Malhas adaptativas para simulação de escoamentos multifásicos

Luzia de Menezes Romanetto

Malhas adaptativas para simulação de escoamentos multifásicos

Luzia de Menezes Romanetto

Orientador: ***Prof. Dr. Fabricio Simeoni de Sousa***

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC/USP como parte dos requisitos para obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

USP - São Carlos
Abril/2014

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

Romanetto, Luzia de Menezes
R758m Malhas adaptativas para simulação de escoamentos multifásicos / Luzia de Menezes Romanetto; orientador Fabricio Simeoni de Sousa. -- São Carlos, 2014.
57 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2014.

1. Malhas dinâmicas. 2. Tratamento de interfaces em movimento. 3. Escoamentos multifásicos. I. Sousa, Fabricio Simeoni de, orient. II. Título.

Agradecimentos

Agradeço ao Bruno pelo apoio e compreensão, e algumas boas ideias também.

Ao Tor pela alegria nos momentos tristes.

Aos meus amigos que sempre apoiaram meus sonhos: Salomão, Martha, Leo, Danilo, Gabs, Fernanda, Isotilia, Roberta, Marcelo, Frodo, Timtim, Laureano, e outros agregados.

Ao ICMC pela estrutura e ao CNPQ pelo apoio financeiro.

Resumo

Simulações de escoamentos multifásicos são de grande interesse em aplicações práticas na indústria, em particular na indústria petrolífera, entre outras. Vários processos dependem do entendimento físico de escoamentos envolvendo iteração com partículas, sedimentação e separação de fluidos. Dos muitos métodos existentes para a simulação dos processos acima descritos, há um crescente interesse no aumento de precisão, o que levou ao desenvolvimento de estratégias que utilizam esquemas de elementos finitos discretizados em malhas dinâmicas e adaptativas, usando uma formulação ALE (do inglês, *Arbitrary Lagrangian-Eulerian*), juntamente com uma representação geométrica da interface. Neste sentido, este trabalho tem o objetivo de estudar e implementar estratégias robustas de controle e adaptação de malhas, em situações onde a malha dinâmica é sujeita a grandes deformações. Uma biblioteca de algoritmos e rotinas foi então desenvolvida para este fim, implementando técnicas de controle e otimização da qualidade dos elementos da malha, técnicas de adaptação da interface entre fluidos com esquemas de conservação de massa, técnicas de mudanças topológicas e preservação de propriedades materiais, além de uma comunicação facilitada destas rotinas com códigos de simulação numérica de escoamentos multifásicos existentes.

Abstract

Multiphase flow simulations are of great interest in practical applications, particularly in the oil industry. Several processes depend on understanding physical aspects of flows with particle interaction, sedimentation and fluid separation. Among the several existing methods to simulate the processes described above, there's a growing interest in achieving higher precision, which led to the development of strategies that use finite element discretization in adaptive, dynamic meshes, using the ALE formatulation along with a geometrical representation of the interface. In this context, this thesis aims to study and implement robust strategies for mesh adaptation, for cases where the dynamic mesh is subject to large deformations. A library of routines and algorithms was developed, implementing mesh elements control and quality optimization techniques, fluid interface adaptation techniques with a mass conservation scheme, topological modifications and material properties preservation techniques, and also a decoupled, simplified communication between these routines with existing multiphase flow numerical simulation code.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO

1

Introdução

Com a grande evolução da Computação nas últimas décadas, a área de pesquisa de Dinâmica de Fluidos Computacional (CFD[1]) atraiu o interesse da indústria e academia. Com as técnicas e tecnologias atuais é possível realizar diversas simulações de fenômenos físicos de fluidos de modo preciso e menos custoso.

Dentre os temas mais estudados em CFD está a simulação de escoamentos multifásicos [31], que envolvem a interação de vários fluidos entre si. De acordo com as características macroscópicas dos fluidos envolvidos, estes podem se misturar ou não. No caso imiscível, onde eles não se misturam, é importante fazer a representação da região que os separa de maneira precisa, pois esta tem grande influência sobre os resultados.

Para discretizar os domínios físicos diversas técnicas em CFD se baseiam na representação destes por malhas. Sabe-se que a precisão dos resultados está intimamente ligada com características geométricas da malha, sendo fundamental acoplar técnicas apropriadas de geração e/ou adaptação de malha para se obter resultados robustos e precisos. Como consequência, a geração e adaptação de malhas também vem sendo estudada, e apresentou uma grande evolução chegando a um estado já bem estabelecido. No caso de simulações de escoamentos multifásicos, exige-se bastante da malha, pois esta além de manter sua qualidade, deve ser capaz de representar regiões de mudança de fase, e manter ao máximo a consistência das informações utilizadas na simulação.

[1] Acrônimo da língua inglesa para *Computational Fluid Dynamics*

Os fenômenos físicos em CFD são modelados inicialmente por leis de conservação, como a conservação de massa, momento, energia e outras. Existem duas abordagens principais para apresentá-las [14]: Euleriana, onde o domínio é fixo; e Lagrangeana, onde o domínio se move com a velocidade do fluido. Uma alternativa a estas é a formulação Lagrangeano-Euleriano Arbitrária (ALE[2]) onde o domínio tem velocidade arbitrária. Essa formulação é vantajosa pois as equações são escritas no referencial da malha, cuja velocidade pode ser determinada tanto para acompanhar o fluido, minimizando os efeitos de difusão numérica, quanto para manter ao máximo sua qualidade.

Um dos métodos mais utilizados na discretização dos modelos de CFD é o método dos elementos finitos (FEM[3]), que tem como base uma discretização do domínio físico dada por uma malha. Trabalhos como [3, 4, 5, 29] demonstram que as características geométricas da malha se relacionam com limitantes inferiores para o erro resultante de simulações pelo FEM. Uma opção bastante atraente para tais simulações é o uso de malhas adaptativas, visto que permitem otimizar as características geométricas em regiões prescritas, melhorando a precisão dos resultados.

A representação computacional de malhas não estruturadas e adaptativas, que são utilizadas neste trabalho, exige uma estrutura de dados complexa, capaz de suportar operações de mudança topológica, como inserção e remoção de elementos, o que possibilita adaptações para reparar a malha. A escolha da estrutura de dados é crucial para garantir bons resultados, e em geral estas são construídas para atender uma necessidade específica, contendo apenas o que é necessário, evitando informações redundantes.

O presente trabalho visou aprimorar a estrutura de dados FEPiC++[4], desenvolvida no ICMC-USP como fruto do trabalho de mestrado [24], com o objetivo de ser uma estrutura de dados genérica para o desenvolvimento de aplicações de elementos finitos. Implementamos uma série de ferramentas sobre esta estrutura para possibilitar o uso de malhas adaptativas. As implementações deste trabalho foram disponibilizadas ao público e estão sendo utilizadas em trabalhos do grupo de pesquisa.

1.1 Objetivo e Aplicações

Nosso objetivo é tornar a biblioteca de malhas adaptativas robusta, capaz de realizar simulações de grande porte de escoamentos multifásicos. Dentre as aplicações vislumbradas estão processos de suspensão e deposição de partículas, decantação de fluidos em gotas e ascensão de bolhas, os quais podem apresentar centenas de estruturas de interesse, podendo ou

[2] Acrônimo na língua inglesa para *Arbitrary Lagrangian Eulerian*
[3] Acrônimo da língua inglesa para *Finite Element Method.*
[4] *Finite Element Programming in C++*

não possuir interfaces deformáveis. Para isso, empregamos modelos de pseudo-física simplificados, visando simular efeitos similares aos encontrados nos escoamentos multifásicos considerados. Com isso, realizamos uma avaliação de desempenho das técnicas de adaptação testadas, bem como ciclos de teste e correção da biblioteca.

A simulação de partículas suspensas em fluidos é um campo amplo, exigindo diferentes métodos de acordo com o tamanho das partículas em relação às características do fluxo. Em uma escala mesoscópica, é necessário considerar o acoplamento de forças das partículas com o fluido, o que é obtido na simulação numérica direta (DNS[5]) calculando diretamente as forças hidrodinâmicas sobre as partículas a partir do fluxo do fluido. Esta simulação direta pode ser a única ferramenta teórica capaz de estudar os fenômenos complexos de interações partícula-parede e partícula-partícula, encontrando aplicações em simulações de erosão fluvial (ilustrado na figura 1.1), transporte lubrificado, sedimentação de hemácias, dentre outras [12, 28].

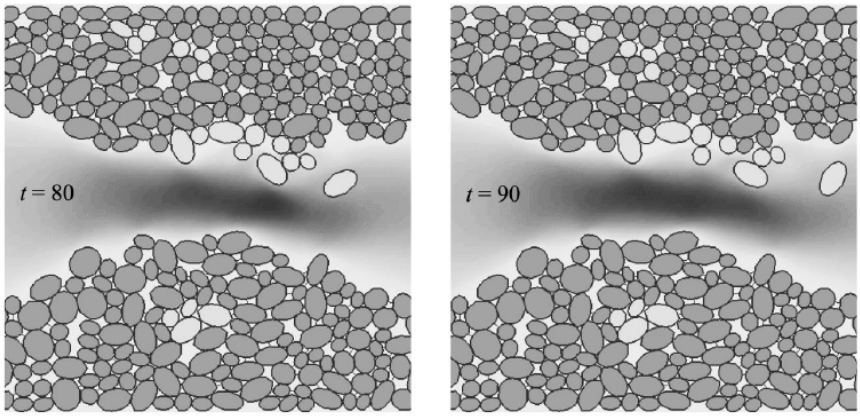


Figura 1.1: Simulação de fluxo erosivo de um particulado cimentado. Obtido de [12].

Outro processo de interesse é o processo de decantação com grande quantidade de gotas, ilustrado na figura 1.2. Neste caso podem ocorrer processos de coalescência, separação de gotas, assim como interações complexas com a parede.

Por fim, o estudo de ascensão de bolhas é um tópico de interesse para entender processos de tranporte de gases em fluidos, como metano gerado no solo de represas hidrelétricas, vapor em fluidos em ebulição, reações químicas, dentre outros [9, 10, 19]. Um aspecto ainda pouco explorado em simulações é o efeito da esteira turbulenta gerado em um trem de bolhas, que aumenta a velocidade terminal em comparação com uma bolha isolada [9].

[5]Acrônimo da língua inglesa para *Direct Numeric Simulation*.

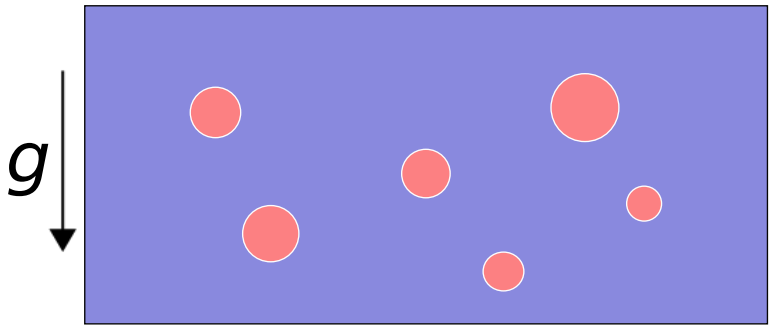


Figura 1.2: Decantação, processo de separação de fluidos por gravidade.

1.2 Organização

Esta dissertação está organizada como segue:

- No Capítulo 2, apresentamos definições básicas sobre malhas, incluindo sua formulação matemática e considerações de implementação da estrutura de dados.
- No Capítulo 3, discutimos sobre métodos de adaptação dinâmica da malha, seus benefícios e considerações de implementação.
- No Capítulo 4, apresentamos os modelos físicos utilizados para as simulações deste trabalho e os fenômenos por ele aproximados.
- No Capítulo 5, mostramos os aspectos computacionais deste trabalho, modelando os experimentos de modo a ressaltar características importantes que esperamos de uma biblioteca de malhas adaptativas.
- No Capítulo 6, apresentamos os resultados obtidos e a avaliação de desempenho realizada, comparando diversas medidas de interesse entre diferentes implementações.
- Finalmente, no Capítulo 7, discutimos as contribuições deste trabalho, assim como orientações para trabalhos futuros.

CAPÍTULO

2

Malhas e estruturas de dados

Neste capítulo revisamos alguns conceitos básicos envolvendo malhas não estruturadas. Serão introduzidas noções relacionadas a tipo de malhas, estruturadas e não estruturadas, células, faces, cantos, vértices, e algumas relações topológicas. Além disso, será apresentada uma pequena documentação da biblioteca de elementos finitos FEPiC++, bem como a documentação da estrutura de dados de malhas contida nesta.

2.1 Malhas

Esta seção é dedicada a apresentar os conceitos fundamentais para contextualizarmos as malhas estudadas neste trabalho.

Definição 1 (Fecho convexo) *Dado um conjunto de pontos $P = \{p_1, p_2, \ldots, p_m\}$ em $\mathbb{R}^n$, o fecho convexo de P, denotado por $\mathcal{CH}(P)$, é o menor subconjunto fechado e convexo de $\mathbb{R}^n$ que contém P. Este subconjunto é unico, e pode ser caracterizado pela seguinte expressão:*

$$\mathcal{CH}(P) = \left\{ x \in \mathbb{R}^n \,\middle|\, x = \sum_{i=1}^{m} \lambda_i p_i, \textit{ tal que } \sum_{i=1}^{m} \lambda_i = 1, \lambda_i \geq 0 \,, i = 1, 2, \ldots, n \right\} \quad (2.1)$$

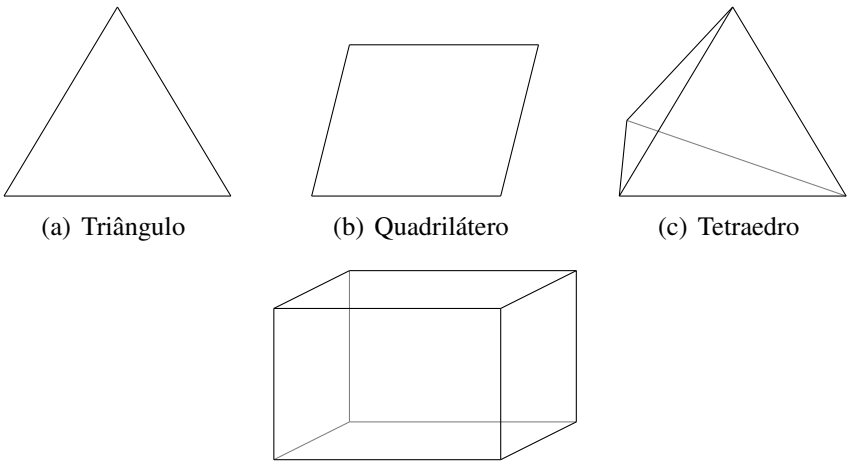
Definição 2 (Simplexo) *O conceito de simplexo é uma generalização dos triângulos em $\mathbb{R}^2$ e tetraedros em $\mathbb{R}^3$ para dimensões arbitrárias. Um simplexo de $\mathbb{R}^n$ é o fecho convexo de $n+1$ pontos independentes de modo afim, isto é, dados os pontos $\{p_1, p_2, \ldots, p_{n+1}\}$, os n*

vetores $\{p_1 - p_2, \ldots, p_1 - p_{n+1}\}$ *são linearmente independentes. Como já dito, um simplexo de* $\mathbb{R}^2$ *é um triângulo e de* $\mathbb{R}^3$ *um tetraedro.*

Definição 3 (Dimensão) *Dado um elemento que contém o conjunto de pontos* $P = \{p_1, p_2, \ldots, p_m\}$*, este é dito de dimensão* n *se existem exatamente* $n + 1$ *pontos independentes de modo afim. No caso dos simplexos, sua dimensão é sempre igual à dimensão do espaço onde este é definido.*

O termo **malha** pode ser definido como coleções de pontos, arestas e polígonos que definem um subconjunto do espaço $\mathbb{R}^n$. Mais do que definir, as malhas são geradas para discretizar os subconjuntos, representada por elementos que guardam informações topológicas e espaciais do objeto que representa, o que nos permite obter informações sobre este de acordo com o tipo de elemento e sua vizinhança. A seguir apresentamos a nomenclatura sobre elementos utilizada neste trabalho.

Definição 4 (Célula) *Os elementos de maior dimensão da malha são chamados de células. Malhas mistas apresentam diferentes tipos de células, enquanto malhas não-mistas apresentam um único tipo. Neste trabalho utilizamos apenas malhas não-mistas, de modo que seu tipo de elemento pode ser utilizado para caracterizá-la. Em* $\mathbb{R}^2$ *as células são polígonos como triângulos, quadriláteros, hexágonos, e outros; em* $\mathbb{R}^3$ *são tetraedros, hexaedros, e outros. Neste trabalho são consideradas apenas células dadas por triângulos, quadriláteros, tetraedros e hexaedros, que são ilustrados na figura 2.1.*



(a) Triângulo (b) Quadrilátero (c) Tetraedro

(d) Hexaedro

Figura 2.1: Ilustração dos tipos de células utilizadas neste trabalho.

Definição 5 (Face) *Dada uma célula* $\mathcal{C}$*, as faces são os elementos de maior dimensão sobre a fronteira de* $\mathcal{C}$*. Para os triângulos e quadriláteros as faces são as arestas; para tetraedros e hexaedros, são triângulos e quadriláteros, respectivamente.*

Definição 6 (Canto) *Os cantos são os elementos que estão contidos na fronteira das células e possuem uma dimensão a menos que as faces. Em $\mathbb{R}^2$ os cantos são dados por vértices (conforme será definido a seguir), e em $\mathbb{R}^3$ os cantos são arestas.*

Definição 7 (Nó) *Os nós são todos os pontos no espaço contidos na malha, podendo se localizar tanto no interior quanto na fronteira dos elementos de maior dimensão. Em geral, no contexto de simulações físicas, a cada nó são associados graus de liberdade dos métodos numéricos empregados sobre a malha.*

Definição 8 (Vértice) *Os vértices são os pontos no espaço que são extremos a todos os elementos de maior dimensão, ou seja, nunca são interiores a outros elementos. Como já dito, em $\mathbb{R}^2$ os vértices coincidem com os cantos.*

Em uma malha os elementos possuem informações espaciais como sua posição, tamanho e forma; e também topológica que fornecem dados sobre a vizinhança do elementos, como a quais elementos este possui interseção. A seguir introduzimos também algumas noções topológicas importantes que caracterizam as relações entre os elementos de uma malha.

Definição 9 (Células adjacentes) *Dadas duas células c_1 e c_2, estas são ditas adjacentes se a interseção entre elas é uma face comum.*

Definição 10 (Incidência) *Dados dois elementos p e q de quaisquer dimensões contidos em uma malha, supondo sem perda de generalidade que p tem dimensão menor que q, p e q são ditos incidentes se p é contido em q.*

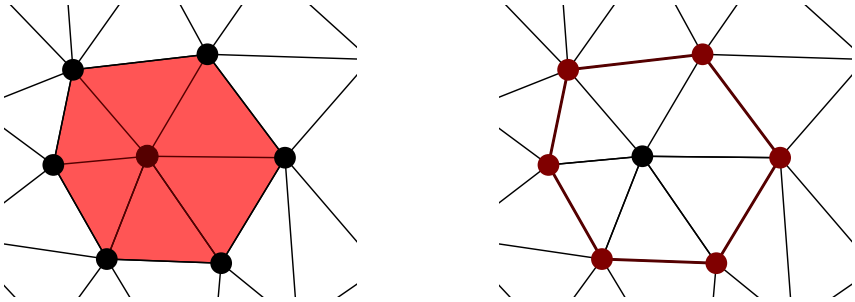
Definição 11 (Estrela) *Dado um elemento p contido em uma malha $\mathcal{M}$, a estrela (**star**, em inglês) de p, denotada por $S(p)$, é conjunto de todos os elementos de $\mathcal{M}$ que contêm p, o que pode ser definido como:*

$$S(p) = \{q \in \mathcal{M} : p \subseteq q\} \tag{2.2}$$

Definição 12 (Elo) *Dado um elemento p contido em uma malha $\mathcal{M}$, o elo (**link** em inglês) de p, denotada por $L(p)$, é o conjunto dos elementos que pertencem à fronteira da estrela de p, o que pode ser defido como:*

$$L(p) = \{q \in \mathcal{M} : q \subseteq \partial S(p)\} \tag{2.3}$$

A estrela e o elo de um ponto são ilustrados na figura 2.2.

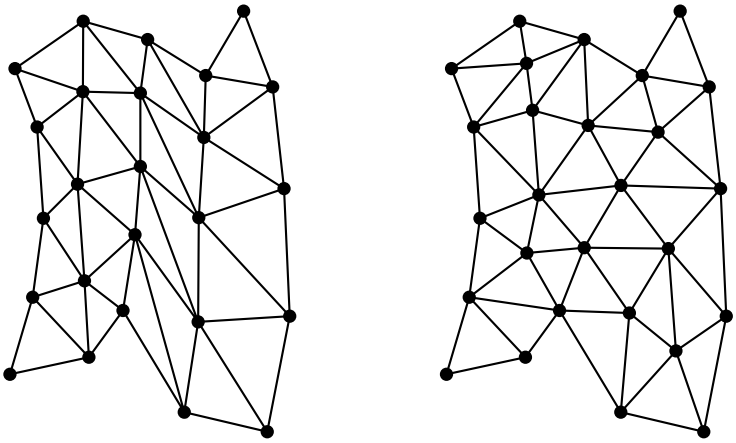


(a) Estrela de um ponto (b) Elo de um ponto

Figura 2.2: Ilustração da estrela e do elo de um ponto.

Fundamentalmente podemos classificar malhas entre dois tipos: malhas estruturadas e não estruturadas. Em malhas estruturadas a topologia é regular, e cada nó interno da malha possui um número constante de elementos em sua estrela, sendo isomorfa a uma grade cartesiana regular, como pode ser visto pelo ilustrado na figura 2.3(a). Tais malhas são bastante simples de se manipular, porém não possuem propriedades necessárias para simulações complexas, como adaptatividade, o que restringe bastante sua área de aplicação.

As malhas não estruturadas têm a seu favor sua adaptação a geometrias complexas, possibilidade de realizar uma gradação de tamanho entre regiões, sem que isso necessariamente prejudique a qualidade dos elementos, e grande generalidade de aplicações possíveis . Porém, ao contrário das malhas estruturadas, sua topologia não é regular, o que demanda uma estrutura de dados mais complexa para manipulá-las, como ilustrado na figura 2.3(b).



(a) Malha estruturada (b) Malha não estruturada

Figura 2.3: Exemplos de malhas estruturadas e não-estruturadas.

Como discutido em [2, 11], para realizar um projeto de implementação de uma estrutura de dados, devem ser levados em conta alguns requisitos, como:

- Eficiência em tempo e memória, de modo que operações de buscas e modificações sobre elementos sejam eficientes e ao mesmo tempo minimizem a demanda por armazenagem em memória.
- Neutralidade da linguagem de programação utilizada.
- Entradas e saídas convenientes, facilitando assim a comunicação entre diferentes módulos da aplicação.
- Fácil extensão e suporte à estrutura, e fácil comunicação entre os módulos que facilitem paralelização.

Porém, integrar e satisfazer tais requisitos muitas vezes pode ser uma tarefa difícil. Por isso, grande parte das estruturas de dados são idealizadas para serem utilizadas em uma determinada aplicação, otimizando os pontos mais impactantes num contexto específico.

A seguir apresentamos a biblioteca de simulação física FEPiC++, que visa ser uma opção para um rápido desenvolvimento de aplicações de simulação em especial para o método de elementos finitos. Dentro desta biblioteca foi implementado um pacote de malhas idealizado para otimizar o acesso a informação.

2.2 Introdução à biblioteca FEPiC++

A biblioteca de elementos finitos *Finite Element Programming in C++* (FEPiC++) foi desenvolvida com o objetivo de sanar uma carência de softwares livres de boa qualidade na área. A biblioteca possibilita rápido desenvolvimento de simulações computacionais com malhas adaptativas, em especial voltada para o método dos elementos finitos, embora possa ser aplicada para a implementação de outros métodos.

Para sua implementação foi utilizada a linguagem C++, que oferece boas opções de encapsulamento, em um paradigma de orientação à objeto. Além disso, foram levadas em consideração funcionalidades da linguagem, como *templates*, tornando-a bastante genérica e maleável a aprimoramentos.

A biblioteca FEPiC++ traz diversos pacotes que possibilitam um rápido desenvolvimento de aplicações de simulação física, que podem ser encontrados no diretório `Fepic` do diretório raiz da biblioteca:

- ***mesh***: Implementação da estrutura de dados de malha idealizada, com funções utilitárias sobre malhas, como checagem de consistência e entrada/saída;
- ***mesh_tools***: Pacote para manipulação de malhas, onde se deu parte dos desenvolvimentos do presente trabalho.

- ***dofhandler***: Pacote para manipulação de graus de liberdade (DOFs[1]) de um modelo físico, administrando a renumeração entre os graus de liberdade e os nós da malha;
- ***shapefunctions***: Pacote com diversas funções de forma (constante, triangular, etc.) para diversos tipos de elementos 2D e 3D, necessárias para métodos de elementos finitos;
- ***quadrature***: Pacote para integração numérica das funções de forma sobre diversos tipos de elementos 2D e 3D;
- ***custom_eigen***: Pacote com utilidades para integração com a biblioteca de álgebra linear Eigen [18].

2.3 Estrutura de dados

A estrutura de dados de malhas implementada na FEPiC++ foi a base para todas as implementações deste projeto. O objetivo do desenvolvimento de uma nova estrutura para o armazenamento e manipulação de malhas foi a otimização do acesso a dados da malha, devido ao grande número de informações acessadas pela implementação no método dos elementos finitos.

A implementação é baseada na estrutura *half-face* [2]. Uma das preocupações para a definição da estrutura foi a minimização do custo computacional necessário para a extração dos dados pelo método numérico. Outro ponto é a generalidade das entidades, possibilitando inserir uma grande variedade de tipos de elementos, e facilitando diversas implementações tanto no escopo da malha, quanto de métodos numéricos.

Na seção 2.3.1 fazemos uma breve descrição das estruturas de dados *Half-Edge* e *Half-Face* utilizadas, e na seção 2.3.2 detalhamos a implementação realizada.

2.3.1 *Half-Edge* e *Half-Face*

As estruturas *Half-Edge* e *Half-Face* são estruturas análogas, sendo a *Half-Edge* para malhas superficiais e a *Half-Face* para malhas volumétricas. O conceito principal, proposto em [22], é centrado nas faces das células. Cada face (seguindo o que foi definido anteriormente, onde em $2D$ as faces são arestas) é divida em duas *half-faces* de orientações opostas, que são ditas irmãs, como ilustrado na figura 2.4. Cada *half-face* conhece sua *half-face* irmã. Assim, com base no conhecimento das *half-faces* irmãs e na orientação das células é possível obter relações de vizinhança entre os elementos, realizar buscas e percorrer a malha.

[1] Do inglês *degree of freedom*

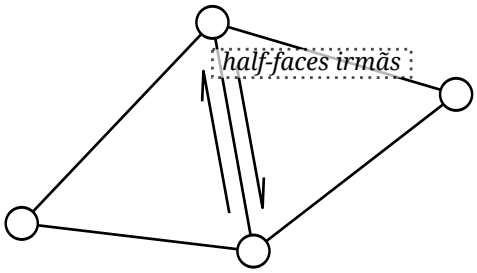


Figura 2.4: Ilustração do conceito principal da estrutura de dados *Half-Face*.

Tais estruturas vem sendo estudadas e usadas amplamente em geometria computacional por sua generalidade e compreensividade, que são condições para que seja possível a representação de malhas dadas em diversos tipos de entrada.

2.3.2 Implementação

Em uma implementação de estrutura de dados, os elementos de uma malha podem ser representados de duas maneiras: explicitamente ou implicitamente. Estruturas implícitas visam minimizar o uso de memória, armazenando apenas os dados indispensáveis sem redundâncias. Apesar de ser econômica no uso de memória, tal tipo de estrutura exige um maior processamento, pois informações necessárias que não estejam armazenadas devem ser extraídas por operações realizadas sobre os dados disponíveis.

Já estruturas explícitas armazenam o máximo de dados necessários, de modo que a maioria das informações são obtidas por acesso direto. Isso exige pouco potencial de processamento, porém um maior uso de memória. Deve-se tomar precauções para que o acesso à memória não se torne o gargalo da aplicação.

A estrutura de dados implementada para a biblioteca FEPiC++ utiliza a representação explícita, inspirada nos trabalhos [7, 20, 21]. Todos os seus elementos são representados por objetos apropriados, que por sua vez armazenam dados necessários para acessar suas adjacências de maneira direta, ou com poucas operações.

Na estrutura de dados da FEPiC++ as entidades de uma malha são classificados em dois tipos: **Células** e **Elementos de Células**. Os **Elementos de Células** são divididos em três tipos: **Faces**, **Cantos** e **Pontos**. Para esta implementação são apenas consideradas malhas não mistas, ou seja, que possuem apenas um tipo de célula. Assim, para cada tipo de malha o tipo de suas **Células**, **Faces**, **Cantos** e **Pontos** são pré determinados de acordo com a tabela 2.1.

As classes que representam os elementos da malha seguem o diagrama de hierarquia de classes apresentado na figura 2.5. Tanto as células como os elementos de células herdam da classe `Labelable` atributos para serem usados como marcadores (*tags*), que na prática

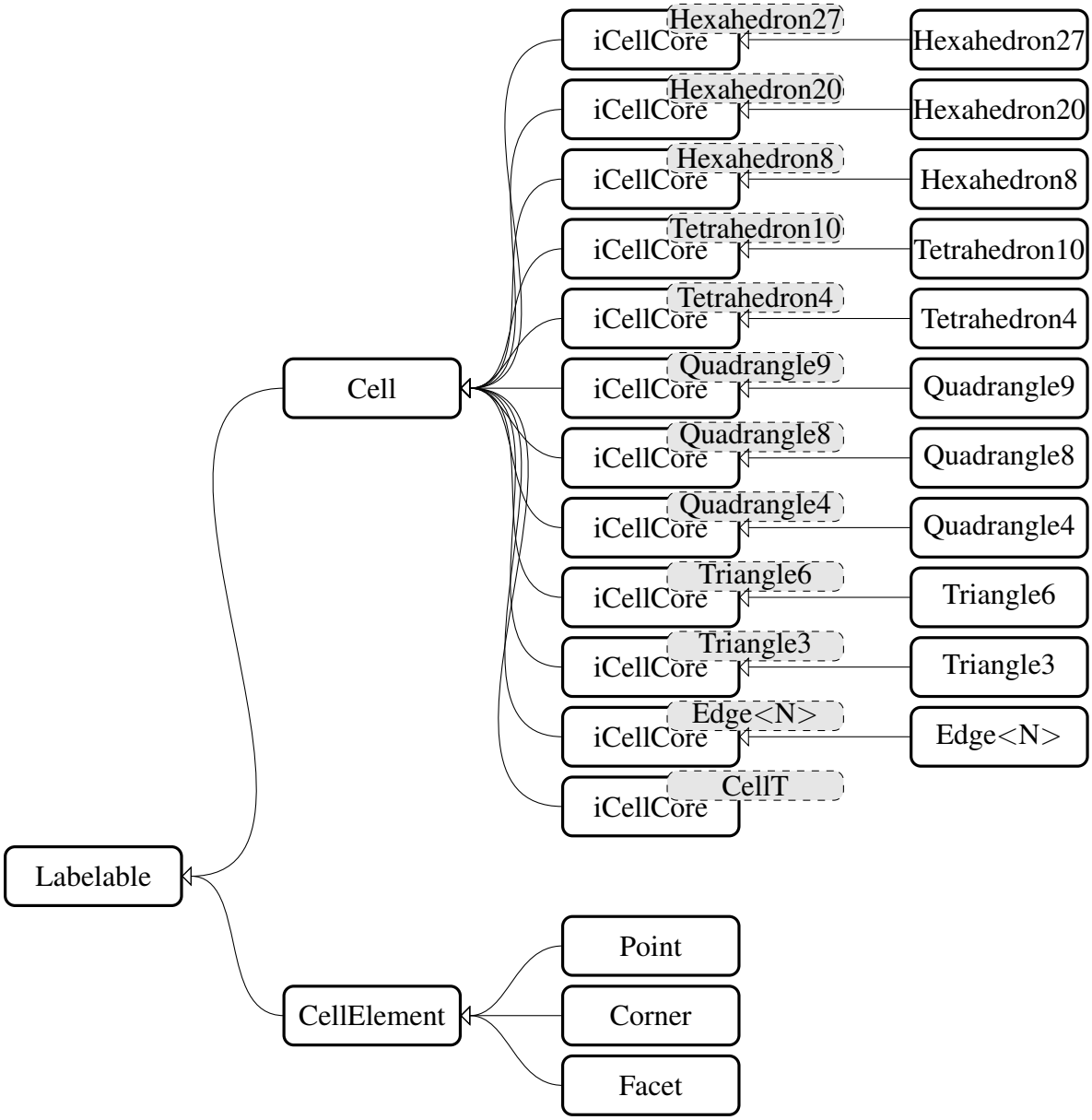


Figura 2.5: Diagrama de hierarquia de classes da estrutura de dados de malha implementada na FEPiC++.

são usados principalmente para armazenar valores que representam propriedades físicas dos elementos.

Os objetos que representam os elementos da malha são armazenados em quatro contêineres para as células, faces, cantos e pontos. Cada objeto, ao ser armazenado em seu devido contêiner, é identificado por um número — que chamaremos de identificador — inteiro, positivo e único para cada elemento. O identificador de cada elemento é associado à posição deste na memória. O contêiner, além de gerenciar os identificadores, também gerencia a disponibilidade de memória para armazenagem dos objetos, podendo realocá-los de acordo com a necessidade.

Dimensão	Célula	Face	Canto	Ponto
1D	Edge<N>	Ponto	Ponto	Ponto
2D	Triangle3	Aresta de dois nós	Ponto	Ponto
2D	Triangle6	Aresta de três nós	Ponto	Ponto
2D	Quadrangle4	Aresta de dois nós	Ponto	Ponto
2D	Quadrangle8	Aresta de três nós	Ponto	Ponto
2D	Quadrangle9	Aresta de três nós	Ponto	Ponto
3D	Tetrahedron4	Triângulo de três nós	Aresta de dois nós	Ponto
3D	Tetrahedron10	Triângulo de seis nós	Aresta de três nós	Ponto
3D	Hexahedron8	Quadrilátero de quatro nós	Aresta de dois nós	Ponto
3D	Hexahedron20	Quadrilátero de oito nós	Aresta de três nós	Ponto
3D	Hexahedron27	Quadrilátero de nove nós	Aresta de três nós	Ponto

Tabela 2.1: Tabela de tipos de elementos suportados em cada tipo de malha.

Cada célula guarda todos os identificadores que referenciam os objetos que representam seus elementos (nós, cantos e faces) tomando como padrão a convenção CGNS [13] de numeração local destes elementos. Além das referências para seus elementos, as células guardam a referência para as células adjacentes e sua posição de incidência, de acordo com a face onde estas células incidem. Deste modo o acesso aos elementos e à vizinhança de uma célula é bastante direto.

Além do acesso direto, a grande vantagem desta estrutura é o fato de acoplar aos elementos da malha conceitos físicos, através do uso dos marcadores herdados da classe `Labelable` (*tags*), facilitando a manipulação e administração de informações importantes ao longo das simulações. As operações de adaptação de malha foram implementadas usando esta funcionalidade para manter, de acordo com regras pré estabelecidas, os marcadores associadas aos elementos da malha ao longo de toda uma simulação. Isso traz ao usuário uma grande facilidade, pois não é necessário que este realize implementações externas à ED de malha para manter e administrar o valor dos marcadores. Contudo, o número de *tags* é limitado, o que teve que ser contornado em algumas aplicações.

2.4 Operações topológicas

Para que uma biblioteca suporte a adaptatividade da malha, são necessárias operações topológicas [6, 15], ou seja, operações que mudam a conectividade dos elementos, como inserção e remoção de elementos. Neste trabalho foram implementadas uma série de operações sobre o pacote de malhas, levando em consideração as informações físicas contidas na malha.

Para realizar simulações de escoamentos multifásicos, é necessário identificar as regiões da malha que correspondem a cada fase. Isso é feito através de marcadores(*tags*) associados

a cada uma destas, como ilustrado na figura 2.6. Em simulações utilizando a formulação Lagrangeana ou ALE a malha se deforma, e muitas vezes são necessárias adaptações para reparar sua qualidade. Tais adaptações podem ser realizadas por meio de operações topológicas, o que gera a criação de novos elementos ou ao colapso de antigos. Durante uma adaptação é necessário um cuidado especial com as *tags*, pois caso contrário as informações sobre as regiões que correspondem a cada fase são perdidas, comprometendo a representação do domínio físico na simulação.

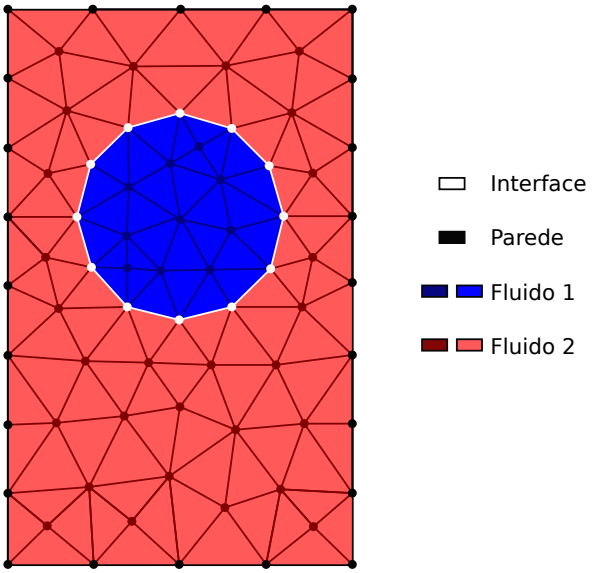


Figura 2.6: Identificação das fases em uma malha.

Em geral, neste trabalho as adaptações envolvendo mudança topológica na malha se baseiam nas seguintes operações: a inserção de um ponto sobre uma aresta, colapso de arestas e *flipping* de aresta. Este último é muito conhecido por ser a base da triangulação de *Delaunay*, e consiste na inversão de uma aresta para melhorar a qualidade dos elementos adjacentes, como mostrado na figura 2.7.

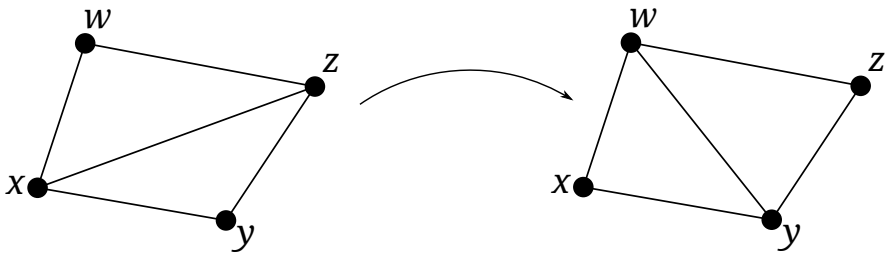


Figura 2.7: Ilustração da operação topológica de *flipping* de aresta.

Ao longo das adaptações dinâmicas existem dois marcadores que devem ser preservados em uma geometria:

- ***Tag* física:** indica de qual material é esta parte da geometria.

- ***Tag* bloqueadora:** diz se esta parte da geometria está bloqueada para distorções causadas pelas adaptações. Se o marcador é *true*, o elemento não pode ser movimentado ou distorcido pela adaptação a menos que esteja interagindo com outros elementos do mesmo material, como será explicado a seguir. Caso contrário, ele é livre para passar por adaptações.

Para que a adaptação seja capaz de preservar tais marcadores é necessário estabelecer critérios para a interação entre geometrias de diferentes marcadores, além de como eles podem ser atribuídos ao longo da geometria.

Para esta aplicação são usados os seguintes critérios para a atribuição dos marcadores:

- ***Tag* física:** Os marcadores físicos não apresentam nenhuma restrição quanto à atribuição aos elementos. Qualquer elemento (nó, face, canto ou célula) pode ter qualquer marcador físico independente dos marcadores físicos dos elementos adjacentes a ele.

- ***Tag* bloqueadora:** Os marcadores de bloqueio indicam se um elemento pode ser movimentado ou distorcido pela adaptação de malha. Por exemplo, quando movimentamos um ponto, movimentamos também as arestas incidentes a este. Para evitar casos como este, as seguintes regras são estabelecidas:

  – uma aresta só pode ser bloqueada se seus dois vértices extremos forem bloqueados;

  – uma célula só pode ser bloqueada se todas as suas arestas forem bloqueadas.

  Qualquer caso diferente destes é permitido. Como nos casos utilizados aqui, realizamos apenas movimentação de pontos, portanto não são necessários outros critérios.

Dados estes critérios, os operadores topológicos foram projetados para que as *tags* sejam preservadas, e operam da seguinte maneira:

- Nas operações de colapso de aresta:

  – se um dos vértices é bloqueado, o colapso ocorrerá sobre este vértice, e sua *tag* será preservada;

  – se ambos os vértices forem bloqueados, o colapso só ocorrerá se ambos possuírem a mesma *tag* física, assim como a aresta entre eles, ou seja, se todos os elementos forem do mesmo material. Se não tiverem a mesma *tag* física o colapso não ocorrerá;

  - se nenhum vértice for bloqueado, o colapso ocorrerá em um ponto sobre a aresta que pode ser determinado por um parâmetro t do método que realiza o colapso, e a *tag* física de menor valor será preservada.

- Nas operações de inserção de pontos sobre aresta:

  - o ponto inserido sempre herda as *tags* física e bloqueadora da aresta sobre a qual é inserido, independente do valor das *tags* das células incidentes.

- Nas operações de *flipping* de aresta:

  - o *flipping* só é permitido em arestas não bloqueadas.

Com tal metodologia é possível realizar adaptações dinâmicas, preservando a representação das fases do domínio físico, como poderá ser visto ao longo deste trabalho.

CAPÍTULO

3

Adaptação de malhas

Neste capítulo são mostradas diversas técnicas para malhas adaptativas. Na seção 3.1 são introduzidas técnicas de adaptação local, que são então descritas nas seções 3.2 e 3.3. Na seção 3.4 descrevemos meios de evitar efeitos colaterais de tais técnicas, como a não conservação de massa entre os fluidos. Na seção 3.5, discutimos a comunicação entre a ferramenta de adaptação e o *solver*.

3.1 Métodos de adaptação de malha

Os métodos de adaptação de malhas podem ser globais ou locais. Em métodos globais, também chamados métodos de remalhamento, toda a malha é afetada pela adaptação e muitas vezes é totalmente reconstruída. Muitos destes métodos possuem a grande vantagem de possuírem garantias teóricas dos limitantes de qualidade sobre a malha resultante, porém são bastante custosos computacionalmente e podem ser suscetíveis a difusão numérica: a introdução de novos elementos exige que as grandezas físicas associadas sejam interpoladas a partir dos elementos anteriores. Reintroduzir elementos na mesma região da malha pode provocar uma difusão artificial destas grandezas, o que é altamente indesejável.

Em métodos locais, a cada passo apenas uma região é adaptada, ocorrendo onde são detectados elementos de má qualidade, resultando na melhoria da malha como um todo. As operações topológicas descritas anteriormente são empregadas para melhorar a qualidade da

malha, e para isso é necessário definir quando e quais operações devem ser empregadas. Podemos operar de duas maneiras:

1. A primeira e mais simples, opera iterando várias vezes sobre os elementos da malha, realizando apenas um tipo de operação a cada passagem pelos elementos que não satisfaçam o critério de qualidade estabelecido. Nesse processo algumas questões devem ser levadas em consideração, pois a escolha da ordem em que as operações são utilizadas afeta o resultado final, e pode gerar alguns comportamentos indesejados, como picos de memória ou perda de precisão numérica.

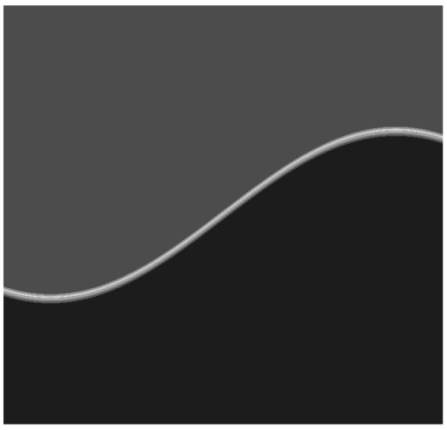
2. Outra maneira de realizar as adaptações são as técnicas de otimização discreta. Tais técnicas operam selecionando ordenadamente vizinhanças de elementos da malha, chamado *cluster*, onde pelo menos um elemento não satisfaz o critério de qualidade, e determina dentre um conjunto de configurações possíveis para este *cluster* aquela que provê a melhor qualidade para aquela vizinhança, sem afetar a qualidade dos elementos externos. Este tipo de técnica tem a vantagem de, a cada operação aplicada, sempre resultar em uma malha de qualidade melhor ou igual à anterior. Contudo, esse processo de otimização pode ficar preso em mínimos locais.

Estimativas de erro do método dos elementos finitos estão relacionadas a características geométricas dos elementos da malha, como tamanho e forma. Os critérios de qualidade podem ser definidos de acordo com as características conhecidas a priori ou a posteriori do comportamento da solução sobre o domínio físico.

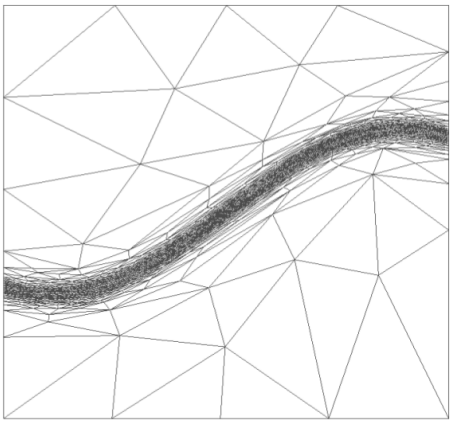
Em regiões onde se deseja maior precisão na solução, elementos menores são desejáveis. Em regiões onde a solução é suave, elementos maiores podem minimizar o custo computacional dos métodos numéricos. Ao mesmo tempo, como mostrado em [29], ângulos muito grandes podem prejudicar a simulação, dificultando a interpolação da solução em casos onde o gradiente é grande.

Em outros casos, medidas que considerem a anisotropia da solução podem ser de bastante utilidade, gerando malhas adequadas para capturar características importantes do comportamento da solução desejada, como mostrado na figura 3.1.

Dadas tais considerações, um bom critério de qualidade deve utilizar tanto o tamanho quanto a forma dos elementos: a forma é crucial para a interpolação da solução sobre o domínio, e se não levarmos em consideração o tamanho, podemos gerar malhas refinadas em demasiado, causando um custo computacional elevado. Neste trabalho utilizamos uma métrica que utiliza ambas as características para o critério de qualidade, como proposto em [8, 11], fazendo os ajustes necessários para adequá-la a cada um dos métodos mostrados nas seções 3.2 e 3.3.



(a) Função que representa uma possível solução.



(b) Malha adaptada para considerar a anisotropia da função.

Figura 3.1: Ilustração do uso de métricas anisotrópicas em adaptação de malhas. Figuras retiradas de [23].

3.2 Método heurístico de adaptação local

Os métodos heurísticos são largamente utilizados por diversos motivos, como: fácil implementação, requerendo poucas ou nenhuma estrutura auxiliar; fácil modificação, dependendo muitas vezes de apenas alguns parâmetros; e apresentam bons resultados, desde que se conheça a dinâmica das adaptações e sua dependência aos parâmetros utilizados, como critérios de qualidade.

Bons resultados podem ser obtidos ao custo de um pouco de experiência por parte do usuário para estabelecer a ordem e os critérios que dominam a adaptação dinâmica. Neste trabalho utilizamos técnicas de refinamento, simplificação e suavização de malhas, onde as operações topológicas são aplicadas de acordo com critérios pré-estabelecidos, iterando pela malha e aplicando uma operação por vez.

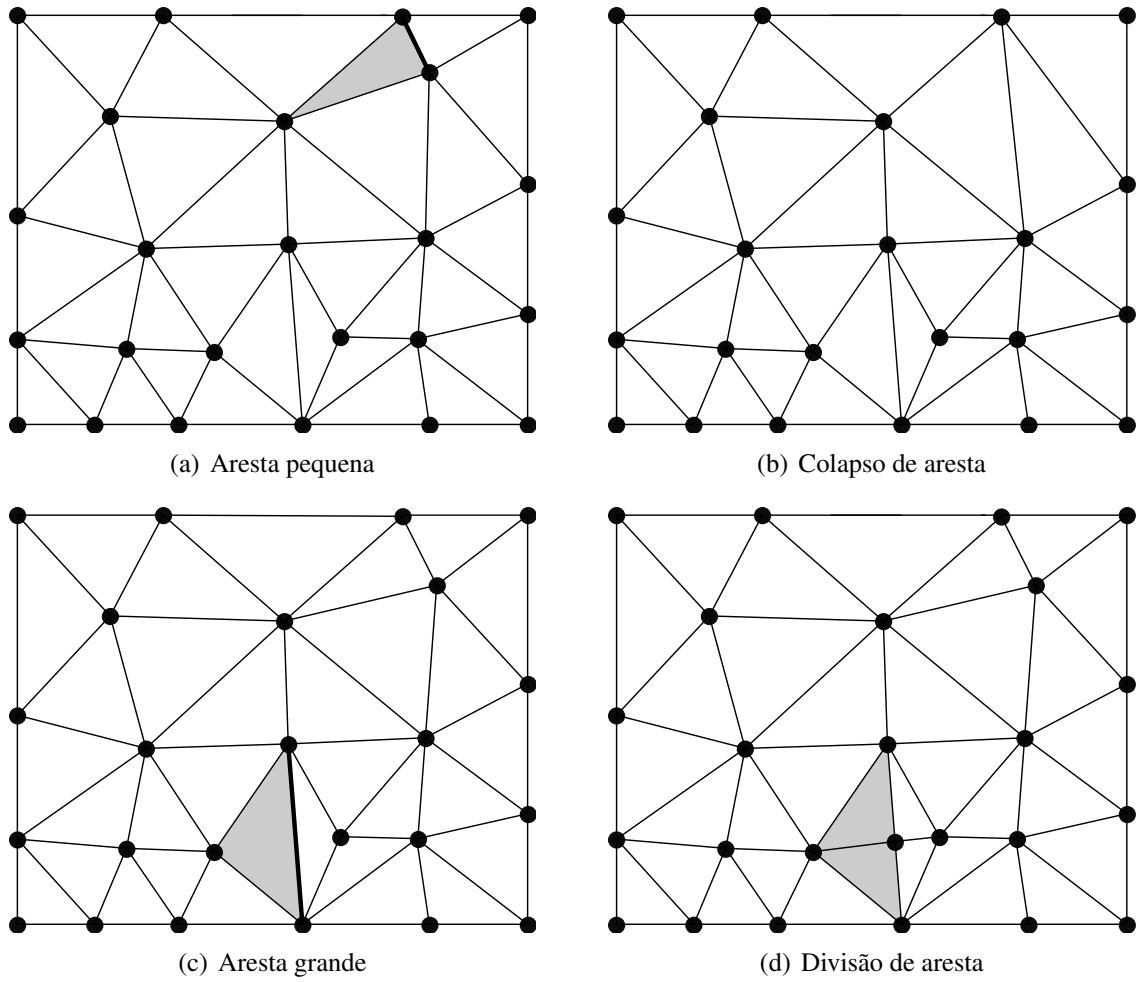
3.2.1 Simplificação e Refinamento de malha

Os métodos de simplificação e refinamento de malha fazem parte das adaptações do tipo h, também chamadas h-adaptação. Tais métodos utilizam operações que alteram o tamanho dos elementos da malha, como colapso e divisão de arestas. Tais operações podem ser usadas de duas maneiras: para controlar o tamanho dos elementos segundo um campo de tamanhos $\delta(\mathbf{x}, t)$ sobre a malha em um tempo t, aplicando divisões em elementos considerados muito grandes, e colapso em elementos considerados pequenos; ou como controle de qualidade com base em medidas que levem em consideração a forma dos elementos.

Neste trabalho utilizamos a primeira abordagem, definindo um campo de tamanhos não uniforme, onde os tamanhos são dados em função da distância do ponto à interface. Dado tal campo $\delta(\mathbf{x}, t)$ definido sobre a malha em um tempo t, como proposto por [11] a seguinte medida adimensional é definida:

$$L_e(t) = \int_e \frac{1}{\delta(\mathbf{x}, t)} dl \tag{3.1}$$

onde $L_e = 1$ é o caso ótimo. Com tal medida, definimos um intervalo aceitável $[L_{min}, L_{max}]$. Fazemos o colapso sobre arestas onde $L_e < L_{min}$, e fazemos a divisão sobre arestas onde $L_e > L_{max}$.



(a) Aresta pequena (b) Colapso de aresta

(c) Aresta grande (d) Divisão de aresta

Figura 3.2: h-adaptação: (a) e (b) ilustram simplificação de malha, (c) e (d) ilustram refinamento de malha.

A figura 3.2 ilustra as estratégias de h-adaptação utilizadas neste trabalho. Considere a mesma malha dada nas figuras 3.2(a) e 3.2(c), sobre a qual definimos um campo de tamanhos não uniforme com valores menores na parte inferior e maiores na parte superior. Na figura 3.2(a) destacamos uma célula contendo uma aresta considerada pequena, sendo co-

lapsada para gerar a malha mostrada em 3.2(b). O caso similar para refinamento é mostrado em 3.2(c), onde mostramos uma célula com uma aresta considerada grande, a qual é então dividida gerando a malha mostrada em 3.2(d).

3.2.2 Suavização *Laplaciana*

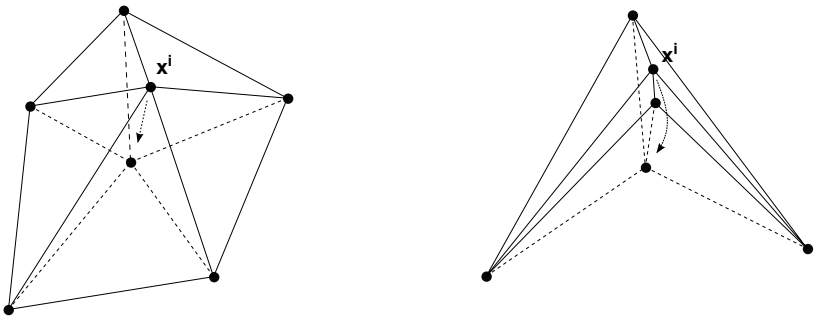
Existem vários métodos de suavização de malhas, que buscam reposicionar os pontos de modo a melhorar a malha segundo algum critério, o que pode ser feito por: processos de otimização, segundo alguma função de custo; simulações físicas, por exemplo modelando a malha como um sistema massa-mola e buscando seu estado de mínima energia; ou baseados em médias, dentre os quais está a *suavização Laplaciana*.

O método de suavização Laplaciana é um dos métodos mais simples e populares de suavização de malha, que consiste em reposicionar os pontos para o baricentro de sua **estrela**, o que pode ser obtido pela média das coordenadas dos pontos contidos no **elo** do ponto, ou seja:

$$\widehat{\mathbf{x}}^i \leftarrow \frac{1}{n^i} \sum_{\mathbf{x}^j \in L(\mathbf{x}^i)} \mathbf{x}^j$$

onde $L(\mathbf{x}^i)$ é o **elo** do pontos $\mathbf{x}^i$, e n^i é o número de pontos contido em $L(\mathbf{x}^i)$.

Se a **estrela** do ponto é côncava, esta média pode levar a um ponto fora da mesma, e o reposicionamento gera elementos inválidos, como ilustrado na figura 3.3(b). Neste caso a adaptação não deve ser realizada.



(a) Reposicionamento no caso convexo.

(b) Reposicionamento no caso côncavo.

Figura 3.3: Exemplos do reposicionamento de um vértice pela suavização Laplaciana, figura retirada de [31].

3.2.3 Algoritmo de adaptação local

O método heurístico de adaptação é dado pelo algoritmo 1 a seguir.

Algoritmo 1 Adaptação de malha pelo método heurístico

Entrada: malha $\mathcal{M}$
  $\mathcal{T} \leftarrow$ CAMPODETAMANHOS($\mathcal{M}$)
  para todo $e \in$ ARESTAS($\mathcal{M}$) **faça**
    $L_e \leftarrow$ COMPRIMENTO(e)
    $L_{min},\ L_{max} \leftarrow$ LIMITANTES($\mathcal{T}$, e)
    se $l < l_{min}$ **então** COLAPSE($\mathcal{M}$, e)
    se $l > l_{max}$ **então** DIVIDA($\mathcal{M}$, e)
  SUAVIZAÇÃOLAPLACIANA($\mathcal{M}$)
  para todo $e \in$ ARESTAS($\mathcal{M}$) **faça**
    se ARESTAÉINVADIDA($\mathcal{M}$, e) **então** FLIP($\mathcal{M}$, e)

Inicialmente realizamos operações de simplificação e refinamento sobre a malha, colapsando as arestas pequenas e dividindo as arestas grandes, de acordo com o critério prescrito pelo campo de tamanhos. As arestas são colapsadas apenas se as regras da operação topológica permitirem, como a conservação de massa, apresentada na seção 3.4. Em seguida, realizamos a suavização laplaciana sobre a malha e, finalmente, realizamos a operação de *flipping* sobre as arestas invadidas [30]. Vale citar que não garantimos que a malha seja de *Delaunay*.

Apesar deste método ser bastante simples e não possuir garantias teóricas de limitantes de qualidade, obtivemos bons resultados, como mostraremos no capítulo 6.

3.3 Otimização discreta por adaptações locais

Definimos a qualidade da malha (ou de uma sub-malha) τ por:

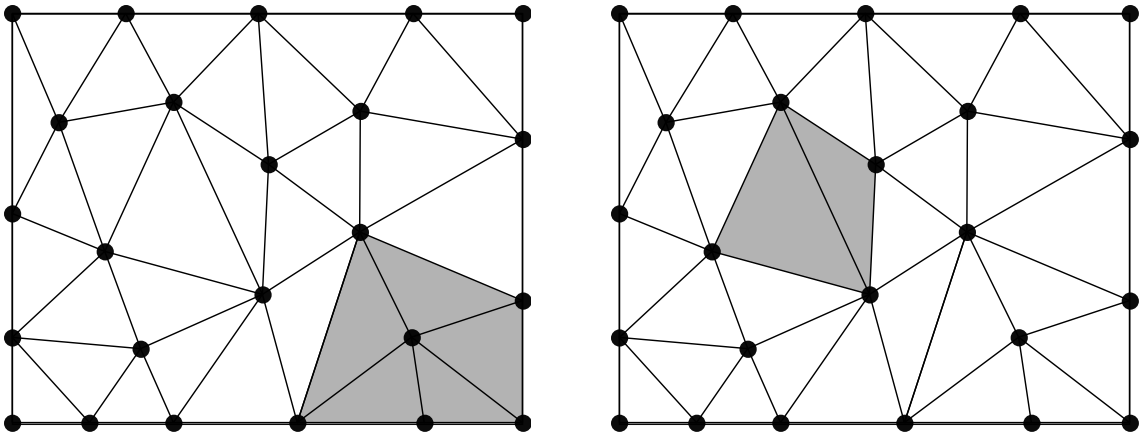
$$Q_\tau = min\{Q_k, k \in \tau\}$$

se $\overline{k}$ é o pior elemento da malha (i.e., $Q_{\overline{k}} \leq Q_k, \forall k$) na atual iteração do algoritmo de adaptação, a sub-malha $M_{\overline{k}}$ formada por $\overline{k}$ e seus vizinhos é avaliada. Nesta sub-malha são consideradas uma série de possíveis configurações topológicas, obtendo a configuração que otimiza a qualidade da sub-malha M_{k^*}. Se $M_{\overline{k}} \neq M_{k^*}$, adaptamos $M_{\overline{k}}$ para M_{k^*}. Este processo garante que a cada passo a malha será de fato otimizada.

As sub-malhas onde aplicamos a adaptação são de dois tipos:

- **Cluster de ponto:** são todas as células que contém um dado ponto, i.e., as células da estrela do ponto.

- **Cluster de aresta:** são as duas células que compartilham uma dada aresta, ou no caso de arestas de bordo a célula que a contém.



(a) Cluster de ponto. (b) Cluster de aresta.

Figura 3.4: Tipos de *cluster*

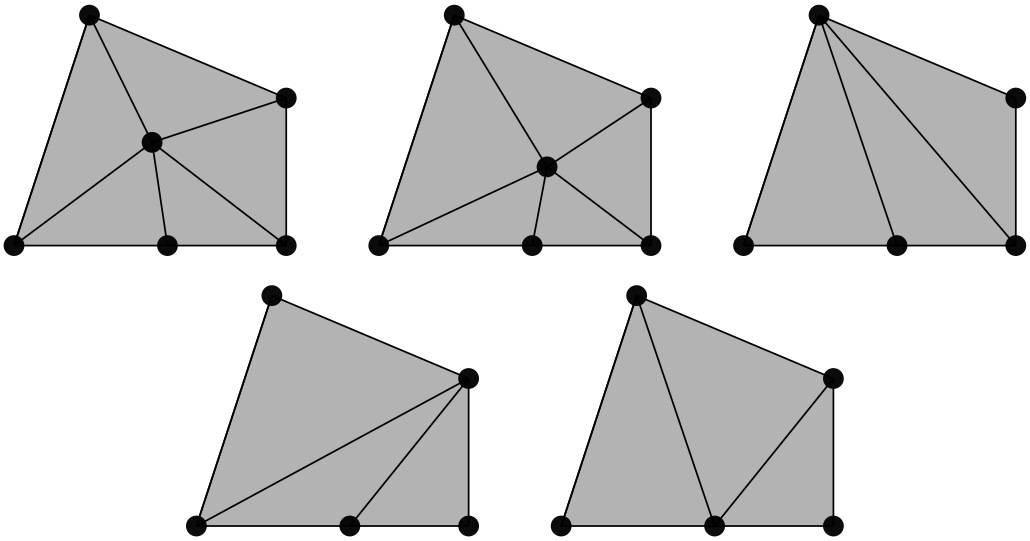


Figura 3.5: Configurações possiveis para *cluster* de ponto.

Cada tipo de *cluster* admite um conjunto de possíveis configurações topológicas onde a qualidade é avaliada para o processo de adaptação. Em um *cluster* de ponto são considerados os casos de colapsar cada aresta incidente ao ponto ou mover o ponto para o centroide de sua estrela, como ilustrado na figura 3.5. Devem-se excluir os casos que gerem elementos invertidos ou degenerados. Em um *cluster* de aresta são considerados os casos de inserir um ponto sobre o centro da aresta, ou realizar o *flip* de aresta, como ilustrado na figura 3.6. Estas operações possibilitam tanto a simplificação quanto o refinamento da malha, sempre garantindo que o processo leve a uma malha otimizada.

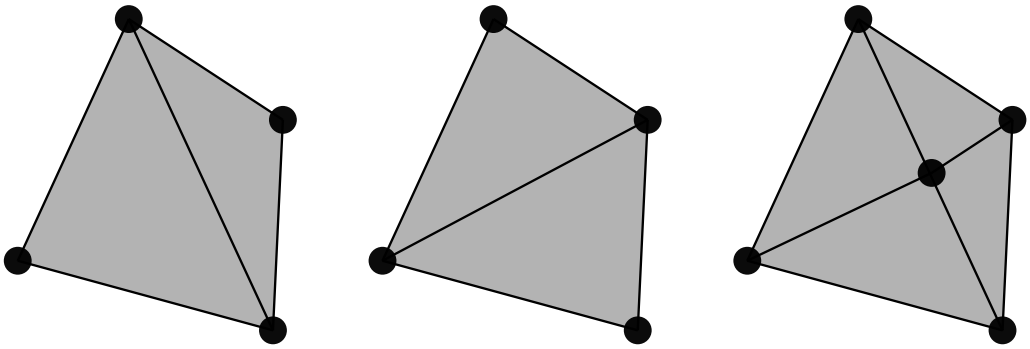


Figura 3.6: Configurações possiveis para *cluster* de aresta.

Para medir a qualidade de cada *cluster*, devemos inicialmente medir a qualidade das células da submalha, dado que o mínimo dentre elas define sua qualidade. Isto deve ser devidamente atualizado a cada adaptação, mantendo a consistência dos dados.

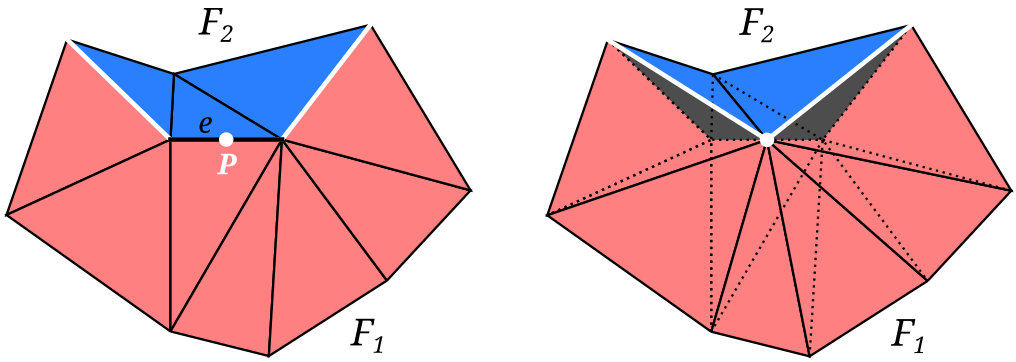
Algoritmo 2 Adaptação de malha pelo método de otimização discreta

Entrada: malha $\mathcal{M}$
Entrada: critério de qualidade Q^*
  $Q \leftarrow$ QUALIDADE($\mathcal{M}$)
  se $Q < Q^*$ **então**
    $L \leftarrow [\,]$
    para todo $p \in$ PONTOS($\mathcal{M}$) **faça** ▷ Adaptação dos *clusters* de ponto
      $\mathcal{C}_p \leftarrow$ CLUSTER(p)
      $Q_p \leftarrow$ QUALIDADE($\mathcal{C}_p$)
      se $Q_p < Q^*$ **então** INSERE($L, \mathcal{C}_p$)
    para todo $\mathcal{C} \in$ ORDENA(L) **faça**
      $\mathcal{C} \leftarrow$ MELHORCONFIGURAÇÃO($\mathcal{C}$)
    $L \leftarrow [\,]$
    para todo $e \in$ ARESTAS($\mathcal{M}$) **faça** ▷ Adaptação dos *clusters* de aresta
      $\mathcal{C}_e \leftarrow$ CLUSTER(e)
      $Q_e \leftarrow$ QUALIDADE($\mathcal{C}_e$)
      se $Q_e < Q^*$ **então** INSERE($L, \mathcal{C}_e$)
    para todo $\mathcal{C} \in$ ORDENA(L) **faça**
      $\mathcal{C} \leftarrow$ MELHORCONFIGURAÇÃO($\mathcal{C}$)

3.4 Conservação de massa

Ao aplicar adaptações de mudança topológica sobre malhas que representam domínios físicos, estas não devem afetar propriedades físicas dos fluidos envolvidos, como por exemplo a quantidade de massa. Porém, quando admitimos o colapso de arestas sobre a interface que separa dois fluidos, tal variação pode ocorrer.

Considere a figura 3.7(a): se a aresta e sobre a interface entre os fluidos 1 e 2 deve ser removida, e é adotada a estratégia de colapsá-la em um ponto P sobre ela, a malha é localmente alterada para a malha mostrada na figura 3.7(b). Visualmente podemos perceber que o colapso causou a transferência da região acinzentada do fluido 2 para o fluido 1.



(a) Malha inicial (b) Colapso sobre o ponto P

Figura 3.7: Transferência de massa causada pelo colapso de aresta.

Uma estratégia para evitar tal efeito é movimentar o ponto P de colapso de modo a balancear a massa entre os dois fluidos, restaurando as quantidades iniciais.

Sejam P_1 e P_2 os pontos extremos de e, e F a região da malha dada pela união de todas as células incidentes a P_1 ou a P_2, i.e.:

$$F = \{t \in \mathcal{M}; P_1 \in t \text{ ou } P_2 \in t\} \tag{3.2}$$

onde F pode ser dividida em duas regiões F_1 e F_2, tal que F_1 são células que pertencem ao fluido 1 e F_2 são as células que pertencem ao fluido 2. Definimos também A_1^i (A_1^f) a área inicial (final) da região F_1 e A_2^i (A_2^f) a área inicial (final) da região F_2, e com isso podemos definir a variação causada pelo colapso como:

$$\Delta A = A_1^f - A_1^i = A_2^i - A_2^f \tag{3.3}$$

a variação de área na região que representa o fluido 1 tem mesma magnitude e sinal contrário à variação na região do fluido 2. Portanto, basta saber a variação em uma das regiões que representam os fluidos, e faremos toda a dedução do método sobre a região 1, sem perda de generalidade.

Seja $\mathbf{E}$ o conjunto de tadas as m arestas e_i contidas em F_1, tal que são opostas a P_1 ou a P_2 em algum triângulo t_k. Sejam os vértices de e_i denotados a_i e b_i, com $b_i = a_{i+1}, i = 1, \ldots, m-1$, ordenados de maneira a preservar a orientação de t_i, como ilustrado na figura 3.8.

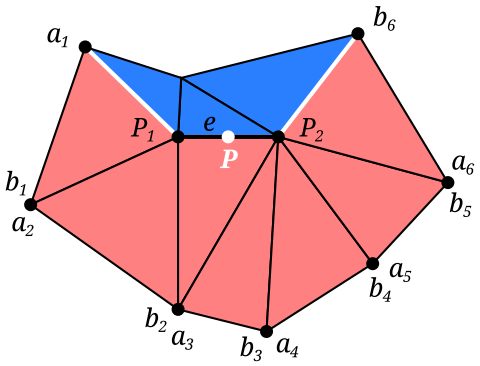


Figura 3.8: Numeração dos elementos para dedução do método de conservação de massa.

Quando a aresta e é colapsada sobre um ponto P, a área final da região pode ser calculada como mostrado na equação (3.4).

$$A_1^f = A(P) = \sum_{e_i \in \mathbf{E}} \frac{d(P, \overline{e_i})|e_i|}{2} \tag{3.4}$$

onde $\overline{e_i}$ é a reta coincidente a e_i, $|e_i|$ o comprimento de e_i e $d(P, r)$ é a função distância entre o ponto P e a reta r.

Tal expressão pode ser vista como um função $A : \mathbb{R}^2 \to \mathbb{R}$, cujas variáveis são as coordenadas do ponto P de colapso. Utilizando A buscamos os pontos P^* onde $A(P^*) = A_1^i$. É possível provar que A tem como curvas de nível retas paralelas ao vetor $\overrightarrow{a_1 b_m}$, e consequentemente gradiente perpendicular a $\overrightarrow{a_1 b_m}$. Logo, o que estamos buscando é uma reta r^* cujo valor seja igual a A_1^i e todos os pontos de r^* serão pontos de conservação de massa.

Para encontrar tal reta basta encontrar um ponto sobre ela, pois já sabemos que $\overrightarrow{a_1 b_m}$ é seu vetor diretor. Para isso partimos do ponto P, e caminhamos na direção $\hat{\mathbf{s}}$ do gradiente de A em sentido contrário ao sinal da variação de massa ΔA. É possível provar que tal ponto está a uma distância igual a $\delta^* = 2\frac{\Delta A}{\|\overrightarrow{a_1 b_m}\|}$ do ponto P.

Podemos assim mover o ponto P para qualquer ponto sobre a reta r mostrada na equação (3.5), e teremos a conservação de massa em ambos os fluidos.

$$r : (P + \delta^* \hat{\mathbf{s}}) + \lambda \overrightarrow{a_1 b_m}, \quad \lambda \in \mathbb{R} \tag{3.5}$$

Este método pode ser adaptado e facilmente aplicado para adaptação de superfícies livres. Tanto no caso de interfaces entre fluidos, quanto em superfícies livres, devemos tomar cuidado para que tal deslocamento não gere células invertidas. Uma opção para isso é buscar a menor perturbação possível, que ocorre quando $\lambda = 0$, logo $P^* = (P + \delta^* \hat{\mathbf{s}})$. Ainda assim podem ocorrer a inversão de elementos, o que deve ser tratado buscando uma posição

válida que pode não existir. Neste caso, a adaptação não deve ocorrer. Tais casos não são frequentes, pois a cada adaptação as variações são pequenas e consequentemente também as perturbações.

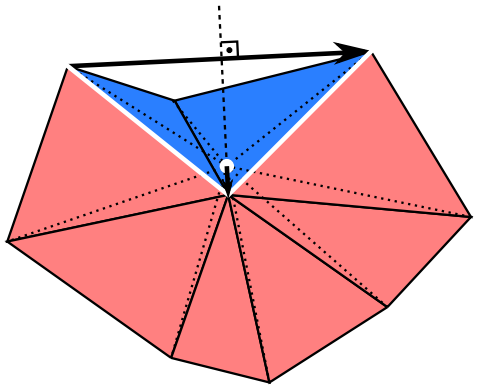


Figura 3.9: Movimentação do ponto colapsado de δ^* unidades na direção perpendicular ao vetor $\overrightarrow{a_1 b_m}$, restaurando a quantidade de massa.

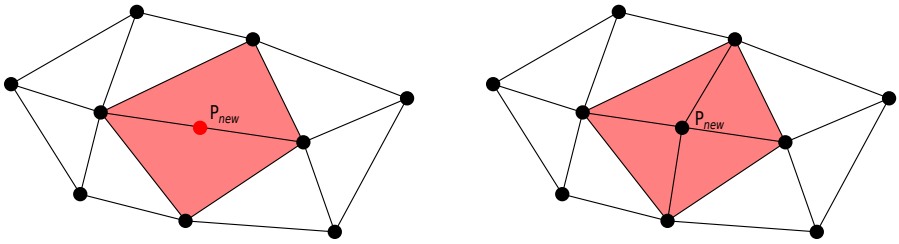
3.5 Comunicação entre a adaptação de malhas e o *solver*

Quando realizamos adaptações de malha, novos nós são inseridos ou movimentados, e é necessário calcular o valor das grandezas físicas nas novas posições ocupadas. Para calcular estas incógnitas é necessário utilizar a configuração topológica da malha anterior à realização da adaptação, para que seja possível interpolar as funções de forma e obter as informações do novo ponto. A biblioteca de malhas é responsável por definir as novas posições, enquanto o cálculo é responsabilidade do *solver*, e para manter estes dois componentes desacoplados precisamos estabelecer uma maneira para que eles se comuniquem durante a adaptação.

Utilizaremos um exemplo onde temos uma cavidade onde será inserido um nó em seu interior. Inicialmente, a malha é como mostrado na figura 3.10(a), e após a inserção a malha passa a ser como na figura 3.10(b). Precisamos calcular o valor das grandezas físicas no ponto P_{new}. O algoritmo para a adaptação é o seguinte:

Algoritmo 3 Adaptação de malha

1. O *solver* detecta que é necessário uma adaptação em uma região da malha.
2. O *solver* invoca a biblioteca de adaptação, passando os identificadores necessários que definem a região onde deve ser realizada a adaptação.
3. A biblioteca de adaptação realiza a modificação na malha, e retorna o identificador do nó gerado ou movimentado.
4. O *solver* continua a simulação com a malha alterada.



(a) Malha inicial. (b) Malha final.

Figura 3.10: Inserção de um nó na malha.

Quando a biblioteca de adaptação retorna o identificador do nó, o *solver* precisa calcular o valor das incógnitas nesta nova posição. Se o *solver* possui uma estrutura de dados própria, ele é capaz de calcular os valores das incógnitas sem problemas no novo ponto, pois sua estrutura não foi modificada pela adaptação, apenas a malha. O problema surge quando o *solver* está acoplado à malha: para fazer o cálculo dos valores das incógnitas no novo ponto o *solver* precisa conhecer a configuração anterior da malha (no nosso exempo dado na figura 3.10(a)), porém no momento após a adaptação o *solver* enxerga a configuração final (no nosso exemplo dado na figura 3.10(b)), onde não é mais possível construir uma interpolação.

O problema gerado é: em que ponto fazer o cálculo das incógnitas na nova posição nodal?

Não podemos fazer este cálculo no *solver* antes de chamar a adaptação, pois o *solver* não conhece a nova posição nodal *a priori*. A biblioteca de adaptação não é responsável pela interpolação e portanto não consegue sozinha determinar os valores. Também, não podemos fazer após a adaptação, pois neste ponto a malha já foi alterada e não faz mais sentido fazer a interpolação baseda na malha.

Uma solução para isso é utilizar um *callback*, onde um objeto recebe uma referência como parâmetro que lhe permite realizar uma chamada externa [1] para outro objeto, responsável por realizar uma determinada operação. Aplicado ao problema em questão, a modificação proposta é que o *solver* passe um ponteiro de função para a biblioteca de adaptação,

[1] *callback* é "chamar de volta" em inglês.

que invoca esta função e armazena sua resposta antes de realizar as modificações, retornando para o *solver* ao final da adaptação as novas posições nodais e suas grandezas físicas já calculadas. Temos os seguintes protótipos para a função de modificação e a função de cálculo de valores nodais:

```
VertexId adaptation(CellId cell_id, EdgeId edge_id,
                    function ptr_func, void *values)

void* Variables(Real *P_new_coords, Cell_id cell_id)
```

O usuário precisa fornecer um ponteiro de função com a assinatura de `function` que implemente os cálculos, e então passá-la para a biblioteca de adaptação em `adaptation`, que não precisa conhecer como realizá-los. Com isso mantemos uma implementação desacoplada dos módulos, o que facilita sua manutenção e extensão.

CAPÍTULO

4

Modelos de pseudo-física

Neste capítulo apresentamos modelos que desenvolvemos para simular comportamentos presentes nas aplicações, para as quais pretendemos empregar os métodos desenvolvidos neste trabalho. Na seção 4.1 apresentamos um modelo de deposição de partículas, no qual temos como objetivo testar a robustez dos métodos simulando grandes quantidades de partículas; na seção 4.2 apresentamos um modelo de ascensão de bolhas, com o qual pretendemos analisar o comportamento dos métodos em simulações com superfícies em deformação.

4.1 Modelo de deposição de partículas

Nosso objetivo é simular um sistema de N partículas sólidas, circulares e imersas em um fluido de densidade ρ_L e viscosidade μ_L. O domínio é discretizado por uma malha de triângulos, sendo a parte ocupada pelo fluido no tempo t denotada por $\Omega_L(t)$, e cada partícula é identificada por sua posição central dada por $X_i(t)$, pelo seu raio r_i, densidade ρ_i e seu domínio é denotado por $\Omega_i(t)$.

O fenômeno é então simulado em função da posição central das partículas, e são utilizadas basicamente 3 forças ativas sobre cada partícula:

- As forças de corpo ativas no sistema são restritas à gravidade, sobre cada partícula.

$$F_{gi} = -\rho_i \pi r_i^2 \ g \ \widehat{\mathbf{j}}, \tag{4.1}$$

- Força de empuxo.

$$F_{ei} = \rho_L \pi r_i^2 \ g \ \widehat{\mathbf{j}} \tag{4.2}$$

- As forças de interação entre o fluido e as partículas são aproximadas considerando suas formas circulares, de modo que o arrasto à viscosidade é dado por:

$$F_{di} = -c_d \ \frac{\rho_L}{2} r_i^2 \ \|V_i\|^2 \ \widehat{V_i} \tag{4.3}$$

onde c_d é o coeficiente de *drag* (arrasto, em inglês) aproximado pelo caso de um cilindro infinito como $c_d = 0.4$.

Desta forma as variáveis do nosso sistema são dadas pela posição e velocidade de cada partícula, denotadas por $\mathbf{X}(t) = (X_1(t), X_2(t), \ldots, X_N(t))$ e $\mathbf{V}(t) = (\frac{dX_1(t)}{dt}, \frac{dX_2(t)}{dt}, \ldots, \frac{dX_N(t)}{dt})$, respectivamente. Suas posições são obtidas pela solução do sistema de equações ordinárias dado na equação (4.4).

$$\begin{cases} \frac{d\mathbf{X}(t)}{dt} &= \mathbf{V}(t) \\ \frac{d\mathbf{V}(t)}{dt} &= \mathbf{Fm}(t) \end{cases} \tag{4.4}$$

onde $\mathbf{Fm}(t) = (\frac{F_1(t)}{m_1}, \frac{F_2(t)}{m_2}, \ldots, \frac{F_N(t)}{m_N})$ e $F_i(t) = F_{gi} + F_{ei} + F_{di}$.

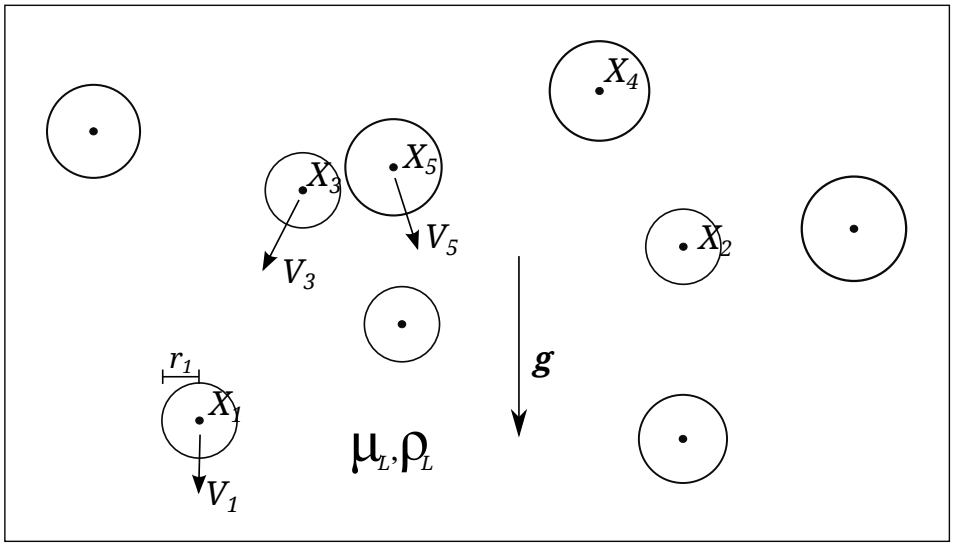


Figura 4.1: Sistema dinâmico simulado.

Consideramos a condição inicial de repouso, $\mathbf{X}(t) = 0$ e $\mathbf{V}(t) = 0$. Para tratar o caso onde onde duas partículas colidem introduzimos uma força adicional quando detectamos a proximidades ou contato entre elas. Consideramos que duas partículas estão em contato quando existir pelo menos uma aresta da malha que conecte as superfícies das partículas.

Neste caso, dentre as arestas que ligam as superfícies, utilizamos o comprimento da menor para estabelecer a seguinte força de repulsão, entre as partículas i e j:

$$F_{ij}^r = -F_{ji}^r = -\xi \frac{1}{e_{min}} \widehat{\mathbf{v}}_{ij}(t) \tag{4.5}$$

onde ξ é um parâmetro obtido experimentalmente que será mostrado a cada simulação, e_{min} é o comprimento da menor aresta, e $\widehat{\mathbf{v}}_{ij}(t)$ é o vetor direção que une os centros das partículas i e j, com sentido de i para j, ou seja, $\widehat{\mathbf{v}}_{ij} = \frac{X_j(t) - X_i(t)}{\|X_j(t) - X_i(t)\|}$. Deste modo obtemos um efeito de colisão praticamente inelástica entre as partículas.

Empregamos também uma força de contato para evitar a penetração da partícula nas paredes, quando existe pelo menos uma aresta que conecte estas superfícies, como ilustrado na figura 4.2. Neste caso, a força possui sempre direção normal à parede, dada por:

$$F_i^w = \xi \frac{1}{e_{min}} \widehat{\mathbf{n}}_w \tag{4.6}$$

onde $\widehat{\mathbf{n}}_w$ é o vetor normal da parede w e os parâmetros ξ e e_{min} são como dados na equação anterior.

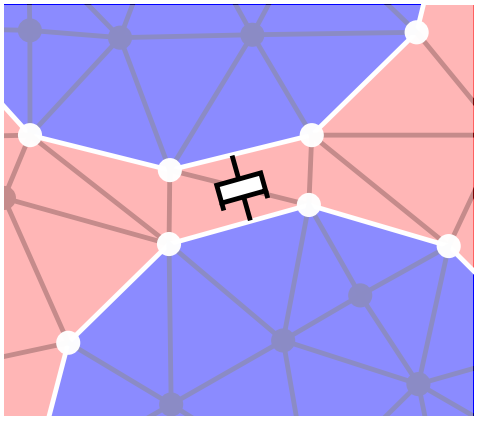


Figura 4.2: Força adicional para evitar que as partículas se sobreponham.

Para resolver as equações que expressam o modelo proposto de deposição de partículas utilizamos o método de Euler explícito, sendo a EDO (4.4) discretizada como mostrado na equação (4.7).

$$\begin{cases} \mathbf{X}(t + \delta t) = \mathbf{X}(t) + \delta t \ \ \mathbf{V}(t) \\ \mathbf{V}(t + \delta t) = \mathbf{V}(t) + \delta t \ \ \mathbf{Fm}(t) \end{cases} \tag{4.7}$$

4.2 Modelo de ascensão de bolhas

Neste modelo, proposto por Cirilo [10], consideramos uma bolha inicialmente esférica de raio r com densidade ρ_B, imersa em um fluido com densidade $\rho_L > \rho_B$, viscosidade μ_L e tensão superficial σ. As características do fluido circundante, e a diferença de densidades entre os dois fluidos determinam se a bolha se deforma ao ascender devido à força de empuxo. A aproximação adotada aqui consiste em considerar a bolha a união de dois semi-elipsoides, com semi-eixos horizontais de comprimento a e diferentes semi-eixos verticais, de tamanho b_1 (inferior) e b_2 (superior). Não consideramos efeitos de difusão, de modo que o volume da bolha deve se manter constante. Uma bolha deformada está ilustrada na figura 4.3.

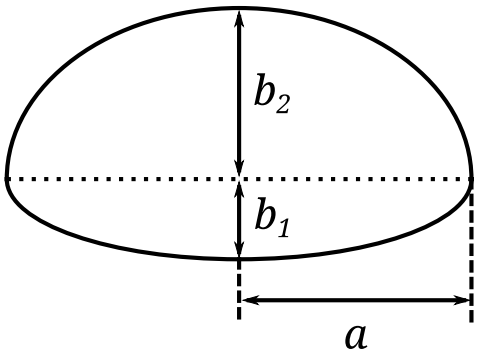


Figura 4.3: Forma variável da bolha, com os semi-eixos indicados.

Definimos a **razão de aspecto** E como uma medida de circularidade da bolha, e o **fator de distorção** γ como uma medida da distorção de uma elipse para uma calota elíptica. Estes valores são definidos para cada instante de tempo nas equações (4.8) e (4.9), respectivamente.

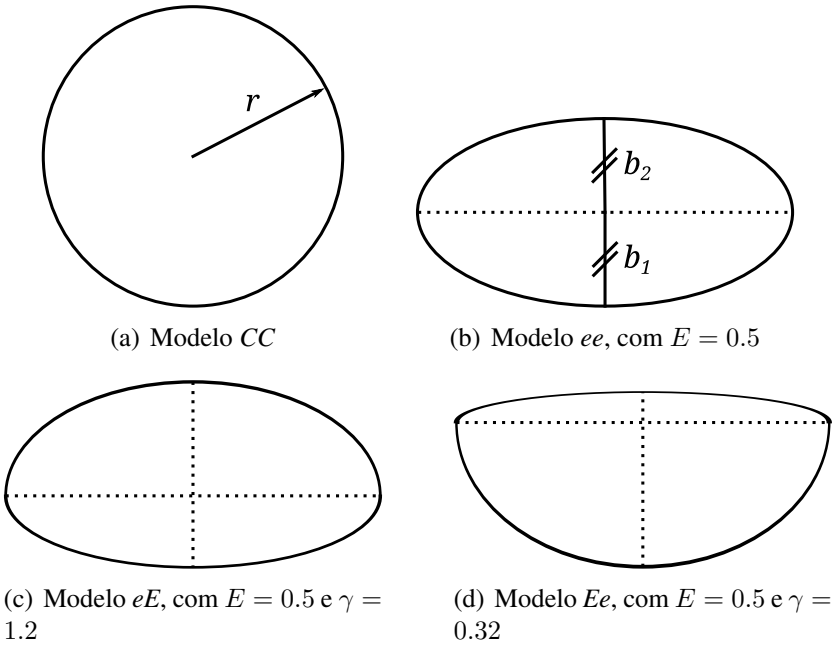
$$E = \frac{b_1 + b_2}{2a} \tag{4.8}$$

$$\gamma = \frac{2b_2}{b_1 + b_2} \tag{4.9}$$

Cirilo considera quatro modelos de deformação que aproximam o comportamento da bolha em diferentes condições[10], resumidos na tabela 4.1. Tais modelos estão ilustrados na figura 4.4.

Modelo	Tipo de deformação	Semi-eixos	Razão de Aspecto	Fator de distorção
CC	Sem deformação	$a = b_1 = b_2$	$E = 1$	$\gamma = 1$
ee	Elíptica	$a > b_1 = b_2$	$E < 1$	$\gamma = 1$
eE	Calota superior	$b_1 \to 0$	$E < 1$	$1 \leq \gamma \leq 2$
Ee	Calota inferior	$b_2 \to 0$	$E < 1$	$0 \leq \gamma \leq 1$

Tabela 4.1: Modelos de bolhas



(a) Modelo *CC*

(b) Modelo *ee*, com $E = 0.5$

(c) Modelo *eE*, com $E = 0.5$ e $\gamma = 1.2$

(d) Modelo *Ee*, com $E = 0.5$ e $\gamma = 0.32$

Figura 4.4: Modelos de bolhas ascendentes. Todas possuem o mesmo volume.

Em experimentos de bolhas de ar imersas em água, a velocidade apresenta uma evolução temporal aproximadamente exponencial, saindo do repouso até uma velocidade limite onde ocorre o balanço entre as forças da gravidade, arrasto e empuxo [9]. Para simplificar a simulação, determinamos uma velocidade terminal V_T e uma constante de tempo τ que definem a velocidade da bolha na direção vertical, dada por

$$v(t) = V_T(1 - \exp(-t/\tau)) \tag{4.10}$$

A evolução temporal da geometria da bolha é controlada pelos parâmetros E e γ, que dependem dos seguintes números adimensionais:

- **Número de Reynolds**: $\mathrm{Re} = \rho_L v(t) L/\mu_L$
- **Número de Morton**: $\mathrm{Mo} = \frac{g_0 \mu_L^4}{\sigma^3} \frac{|\rho_L - \rho_B|}{\rho_L^2}$
- **Número de Tadaki**: $\mathrm{Ta} = \mathrm{Re}\,\mathrm{Mo}^{0.23}$

onde L é um parâmetro dimensional da bolha, como por exemplo seu diâmetro horizontal $2a$.

A fórmula da razão de aspecto E teve sua correlação com Ta obtida por experimentos [9], dado na equação (4.11). O fator de distorção γ, dado na equação (4.12), foi proposto de

forma *ad-hoc*, de modo que varie suavemente na escala de tempo de simulação [10].

$$E(\mathrm{Ta}) = \begin{cases} 1 & \mathrm{Ta} < 0.3 \\ (0.77 + 0.24\tanh(1.9(0.4 - \log_{10}\mathrm{Ta})))^2 & 0.3 < \mathrm{Ta} < 20 \\ 0.3 & \mathrm{Ta} > 20 \end{cases} \tag{4.11}$$

$$\gamma(t, \mathrm{Ta}) = \begin{cases} 1 & \text{em modelos } \textit{CC} \text{ e } \textit{ee} \\ 2 - \exp(-(t/\tau)\,\mathrm{Ta}) & \text{no modelo } \textit{eE} \\ \exp(-(t/\tau)\,\mathrm{Ta}) & \text{no modelo } \textit{Ee} \end{cases} \tag{4.12}$$

Para garantir a conservação de volume da bolha, devemos alterar a dimensão a conforme a equação (4.13). Os semi-eixos verticais b_1 e b_2 são dados conforme as equações (4.14) e (4.15), obtidas a partir de (4.8) e (4.9).

$$a = rE(\mathrm{Ta})^{-1/3} \tag{4.13}$$

$$b_1 = aE(\mathrm{Ta})(2 - \gamma(t, \mathrm{Ta})) \tag{4.14}$$

$$b_2 = aE(\mathrm{Ta})\gamma(t, \mathrm{Ta}) \tag{4.15}$$

CAPÍTULO

5

Aspectos computacionais

Neste capítulo apresentamos as implementações feitas, descrevendo detalhadamente as entradas e saídas, a identificação e marcação dos domínios da malha. A seguir são apresentados os parâmetros que dominam a adaptação, e como estes devem ser definidos para obter bons resultados.

5.1 Implementação

As operações topológicas descritas na seção 2.4 foram implementadas na biblioteca FEPiC++ dentro do pacote $mesh\_tools$, o que tornou possível o uso de malhas adaptativas com a biblioteca. Para simular os modelos físicos descritos no capítulo anterior, desenvolvemos uma aplicação na linguagem C++ utilizando a biblioteca FEPiC++ e sua estrutura de dados de malha.

A malha inicial é gerada utilizando o software gerador de malhas GMSH [17], que recebe como entrada uma geometria que define o domínio dado por sua fronteira, número de partículas e seus respectivos centros e raios. Tal domínio é composto por quatro partes: parede, interface entre fases, domínio interno e domínio externo, como ilustrado na figura 5.1. O gerador de malhas emite uma malha inicial de boa qualidade produzida por um algoritmo de avanço de fronteira [16], e as células recebem uma marcação correspondente à parte a qual pertencem. Esta marcação será utilizada para gerar as *tags* internas da aplicação como explicado a seguir.

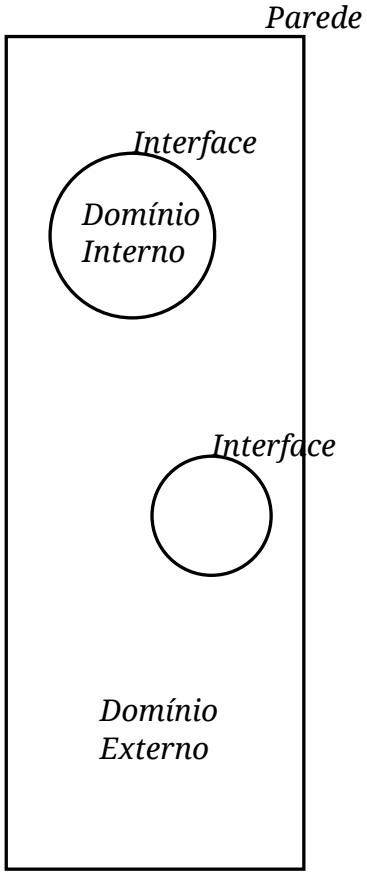


Figura 5.1: Geometria que define o domínio de entrada.

5.1.1 Pré-processamento e definições de *tags* físicas e bloqueadoras

As operações topológicas se baseiam em duas *tags* para manter as informações da simulação e preservar as superfícies: *tags* física e bloqueadora. A motivação da *tag* física é fornecer um identificador distinto para materiais distintos, e está limitada a 256 valores únicos na estrutura de malha da FEPiC++, dado que é armazenada em um único byte por eficiência de memória.

Contudo, gostaríamos que cada bolha ou partícula possuísse uma *tag* física distinta (por simplicidade chamaremos as regiões de bolhas ou partículas como estrutura), para que as operações de adaptação de malha considerem inválido colapsar uma aresta que conecta interfaces de estruturas distintas. Para evitar uma limitação artificial no número de estruturas causado pelo limite de valores da *tag* física, empregamos uma terceira *tag* para identificar a estrutura ao qual o elemento pertence.

A atribuição destas *tags* internas exige que o usuário marque as características de interesse nas arestas e células da malha de entrada, para que a aplicação seja capaz de identificar as

estruturas do domínio físico. Os elementos da malha de entrada podem possuir seis valores como *tags* físicas:

- **Interior**: indica se a célula ou aresta está no interior de uma estrutura;
- **Superfície**: indica se a aresta está na superfície de uma estrutura, isto é, na interface entre fases;
- **Externa**: indica se a célula ou aresta é externa a uma estrutura;
- **Parede**: indica se a aresta é de parede;
- **Bloqueada**: indica que a aresta ou célula é bloqueada;
- **Central**: indica a célula inicial de uma estrutura para a busca em largura, explicado a seguir.

Com estas informações, a aplicação atribui as *tags* identificadoras correspondentes às entradas, e a *tag* bloqueadora como **true** a todos os elementos com *tags* físicas dentre as *tags* bloqueadas. Para aplicar a *tag* identificadora, para cada elemento marcado como central, iniciamos uma busca em largura no grafo de arestas que termina ao se alcançar as arestas de superfície. Os elementos encontrados nesta busca compõem a região conexa de uma estrutura, e então recebem a mesma *tag* identificadora. Os elementos de parede e pertencentes ao fluido externo recebem uma *tag* identificadora própria.

Esta forma de marcação é a mais eficiente para um bom funcionamento das estratégias de preservação de informação que foram implementadas nas operações topológicas deste trabalho.

Após a marcação das estruturas a aplicação calcula o centro e os raios de cada uma, inicializando os graus de liberdade das simulações dinâmicas, com as posições centrais calculadas e velocidade de repouso.

5.1.2 Critérios de adaptação

Os métodos de adaptação de malhas são governados pelos critérios escolhidos, que ditam o que se espera dos elementos da malha. Nesta seção apresentamos uma breve discussão sobre pontos que devemos considerar ao definir tais critérios.

Qualidade de malhas de triângulos

Uma medida de qualidade pode considerar a forma e/ou o tamanho dos elementos, estando ambos intimamente ligadas com limitantes de erros dos métodos utilizados.

Neste trabalho utilizamos duas medidas, uma para cada tipo de adaptação. Para o método de adaptações heurísticas utilizamos como medida a razão média do triângulo, dada por:

$$Q_T = 4\sqrt{3}\frac{A_T}{(\sum_{i=1}^{3}(l(e_i))^2)} \tag{5.1}$$

onde A_T é a área do triângulo T, e $l(e_i)$ é o comprimento da aresta e_i. Tal medida varia no intervalo $[0, 1]$, e apenas considera a forma do elemento, pois o método heurístico usa um campo de tamanhos para controlar o tamanho dos elementos. Para o método de otimização discreta utilizamos uma medida que considera tanto a forma quanto o tamanho dos elementos, dada por:

$$Q_T = 20.78\frac{V_T}{P_T}F\left(\frac{h_T}{h_T^*}\right) \tag{5.2}$$

sendo que:

$$F(x) = m(x)^\beta\left[2 - m(x)^\beta\right] \tag{5.3}$$

onde $m(x) = \min\{x, \frac{1}{x}\}$, e o parâmetro β governa a influência de F dando uma ponderação de importância entre a forma e a escala de tamanhos na qualidade dos elementos. Para valores grandes de β a forma é menos considerada em relação aos tamanhos, e o contrário ocorre para valores de β próximos de zero. Em [8] recomenda-se o uso de $\beta = 3$ para se obter uma boa relação entre a forma e o campo de tamanhos na adaptação. O valor de h_T^* é obtido pela interpolação do campo de tamanhos sobre o triângulo T.

Campos de tamanhos

Os campos de tamanhos desempenham um papel importante na adaptação, pois determinam quais regiões serão mais ou menos refinadas. Definir um campo de tamanhos de modo eficiente é um desafio, pois em simulações com superfícies em deformação, o campo deve ser capaz de acompanhá-las, e devem ser atualizados a cada passo de tempo.

Considere por exemplo o caso de duas partículas muito próximas. Se o campo possui um tamanho mínimo e as partículas se aproximam a uma distância muito menor que tal tamanho, a adaptação será incapaz de manter elementos de boa forma na região entre as partículas, ou tentará colapsar arestas, caso isso seja permitido. Deste modo, o campo deve se ajustar à simulação para que seja possível manter bons elementos em condições extremas.

Como os pontos da malha se movem de maneiras distintas, o campo também não deve ser definido por eles. Existem várias formas de definir tais campos: muito autores utilizam estimativas *a posteriori* do erro de elementos finitos [1, 8, 23], sendo uma abordagem bastante atraente por ser capaz de captar regiões de interesse e anisotropia da solução mesmo

que afastadas de superfícies; outros se baseiam na proximidade a superfícies e na sua curvatura [25, 26, 27]. O intuito é sempre de captar regiões de maior interesse na simulação, seja por maiores variações da solução, seja por captar fenômenos críticos.

Neste trabalho utilizamos um campo de tamanhos baseado na distância à superfície das estruturas, dado por:

$$\delta(\mathbf{x}) = L_{min} + (L_{max} - L_{min})\frac{d(\mathbf{x})}{L_{range}} \tag{5.4}$$

onde L_{range} é uma distância de influência do campo, L_{min} e L_{max} são os tamanhos mínimo e máximo de aresta, respectivamente, e d é uma função de distância dada por $d(x) = \min\{L_{range}, d_{min}(x)\}$, com d_{min} é a distância mínima às superfícies de interesse. Tal campo foi utilizado por se adequar às simulações propostas, e L_{min} é escolhido de modo que as forças de amortecimento de arestas sejam capazes de impedir a sobreposição das estruturas, e mantenha um refinamento satisfatório perto das interfaces.

5.1.3 Iteração

Ao longo do tempo os nós contidos no $\Omega_i(t)$ são movimentados rigidamente com a velocidade da estrutura, e os nós em $\Omega_L(t)$ são mantidos estáticos, a menos quando movidos como efeito da suavização Laplaciana. Esta diferença de movimento causa distorções que demandam o uso de adaptações, e a cada N_{mesh} passos de tempo a malha é testada para avaliar a necessidade de adaptações. Nossa aplicação opera segundo o seguinte algoritmo:

Algoritmo 4 Algoritmo de iteração da simulação dinâmica.

Entrada: malha inicial $\mathcal{M}_0$
Entrada: passo de tempo δt
Entrada: tempo final t_{final}
Entrada: número de passos onde a malha é testada N_{mesh}
Entrada: critério de qualidade Q^*

```
i ← 0
enquanto i · δt < t_final faça
    X_{i+1} ← f(X_i, M_i, t_i, δt)        ▷ Cálculo das novas posições dos pontos
    M_{i+1} ← MOVA(M_i, X_{i+1})
    se mod (i, N_mesh) = 0 então
        Q ← QUALIDADE(M_{i+1})
        se Q < Q* então ADAPTE(M_{i+1})
    i ← i + 1
```

CAPÍTULO

6

Simulações e resultados

Neste capítulo apresentamos os resultados obtidos ao simular os modelos apresentados, que foram implementados usando a estrutura de dados de malha da biblioteca FEPiC++, para testar a robustez da metodologia desenvolvida e dos métodos de adaptação para tratar grandes deformações. As simulações foram realizadas no *cluster* computacional do LMACC, em um servidor com 8 núcleos Intel Xeon E5345 de 2.33 GHz e 16GB de memória RAM.

6.1 Simulação de deposição de partículas

Nesta seção apresentamos os resultados ao simular o modelo de deposição de partículas. O domínio é uma geometria não retangular similar a um funil, mostrado na figura 6.1. As partículas são distribuidas aleatoriamente sobre a região superior do domínio de modo a não se interceptarem, com raios variando entre 1.2 e 2.0, e se depositam de forma compacta na porção estreita inferior.

A simulação então é realizada de acordo com o modelo mostrado na seção 4.1, com uso dos seguintes parâmetros:

$$
\begin{aligned}
\delta t &= 10^{-3}; \\
t_{final} &= 200; \\
N_{mesh} &= 10; \\
g &= 1; \\
\rho_L &= 1.0; \\
\rho_F &= 0.5; \\
\xi_L &= 2.5;
\end{aligned}
$$

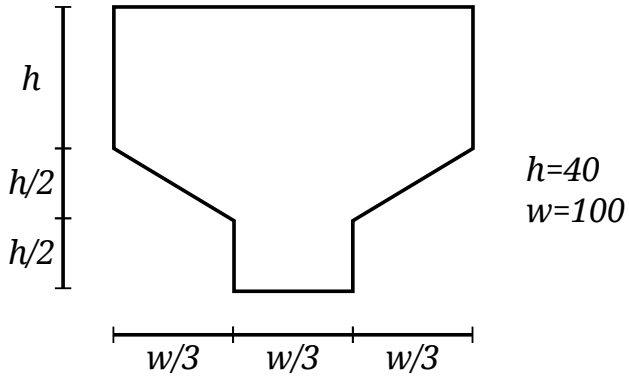


Figura 6.1: Domínio da simulação de deposição de partículas.

Partindo da malha inicial mostrada na figura 6.2 aplicamos o algoritmo 4, utilizando os dois métodos de adaptação dinâmica apresentados. Em ambas as adaptações utilizamos um campo de tamanhos dado em função da distância às bolhas como mostrado na seção 5.1.2, onde $L_{min} = 0.333$, $L_{max} = 1.332$ e $L_{range} = 3.330$.

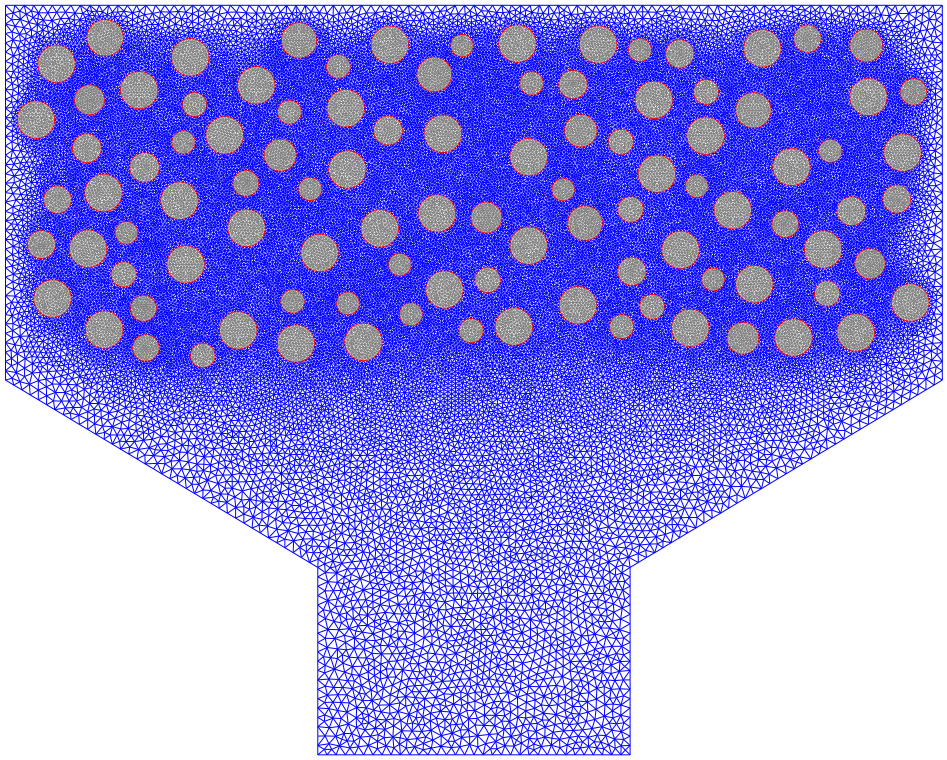


Figura 6.2: Malha inicial da simulação de deposição de partículas, com 87488 células.

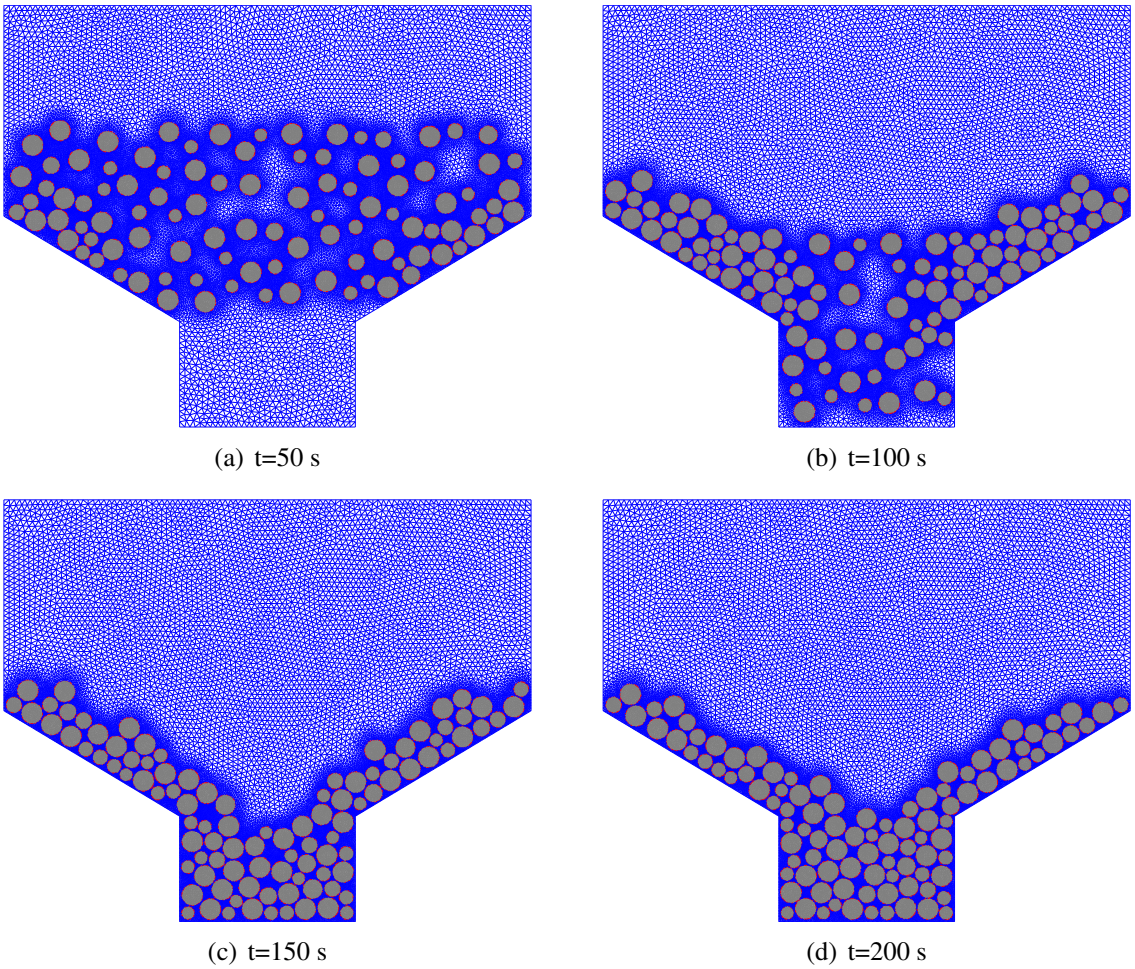
6.1.1 Resultados com o algoritmo de adaptações heurísticas

Aplicamos o método de adaptações heurísticas no algoritmo de evolução temporal, e analisamos como este se comporta de acordo com a variação do parametro Q^* que domina o critério de adaptação, e qual o custo computacional de um critério mais rígido. Foram realizadas 10 rodadas para cada valor de Q^* utilizado para obtermos a variação do tempo de execução. A figura 6.3 ilustra instantes de uma rodada da simulação.

O método é capaz de satisfazer os critérios de qualidade impostos, como pode ser visto na Figura 6.4, que mostra a qualidade mínima dentre os elementos da malha nos últimos 10 segundos de simulação para uma rodada. Nota-se que quando a qualidade se aproxima do limiar estabelecido a adaptação é capaz de reparar a malha, aumentando o mínimo dentre as qualidades dos elementos.

O número de adaptações realizadas é proporcional ao critério de qualidade no intervalo analisado, como mostrado na tabela 6.1. Nota-se que o número de operações realizadas - divisões, colapsos e *flips* - também aumenta de forma proporcional, mas não na mesma razão. Isto indica que embora seja necessário invocar a adaptação mais vezes para satisfazer a um critério mais restrito, são necessárias relativamente menos operações extras para tal.

Dado que o número de operações não varia, percebemos um tempo de execução entre as rodadas com baixa variação relativa, como mostrado na tabela 6.2. Nota-se um aumento no



(a) t=50 s (b) t=100 s

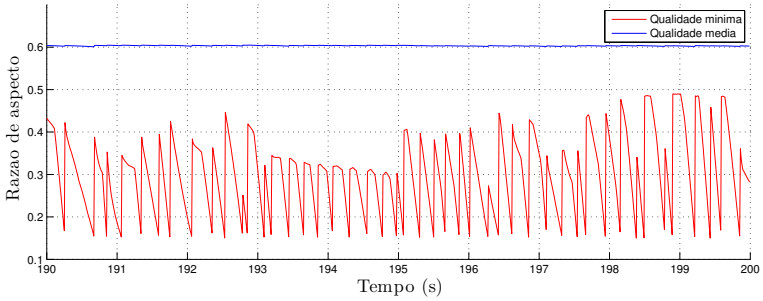
(c) t=150 s (d) t=200 s

Figura 6.3: Simulação de deposição de partículas.

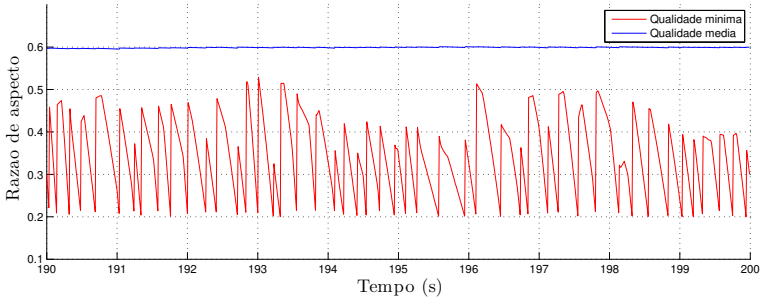
Qualidade	Adaptações	Colapsos	Divisões	*Flips*
$Q^* = 0.15$	1629	298917	283175	482826
$Q^* = 0.20$	1878	303903	288035	509059
$Q^* = 0.25$	2117	308462	292609	529029
$Q^* = 0.30$	2551	324438	308529	572680

Tabela 6.1: Número de operações em função da qualidade imposta. Dado que o algoritmo é determinístico, não existem variações entre as rodadas.

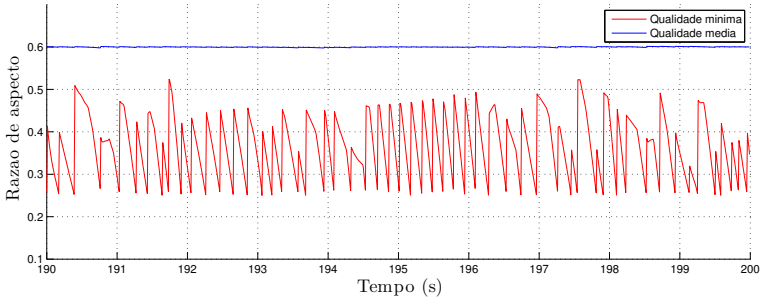
tempo de execução de adaptações proporcional ao critério de qualidade, a uma razão menor do que o aumento de qualidade. Também, nota-se que o tempo de adaptação tem pouco impacto sobre o tempo total, dominado por outras operações como cálculo de qualidade. Para simulações físicas realistas, onde é necessário realizar operações de álgebra linear custosas, o tempo de adaptação não seria expressivo mesmo para um critério de qualidade alto.



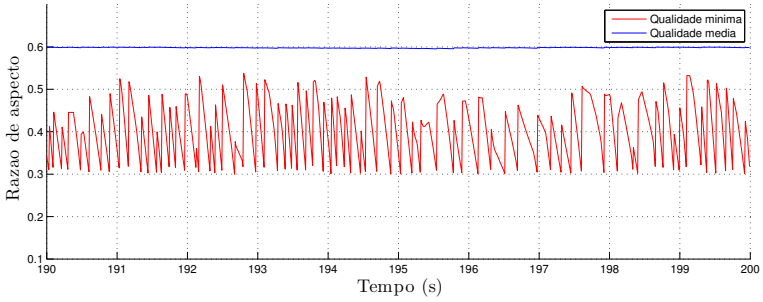
(a) $Q^* = 0.15$



(b) $Q^* = 0.20$



(c) $Q^* = 0.25$



(d) $Q^* = 0.30$

Figura 6.4: Qualidades mínima e média da malha nos últimos 10 segundos simulados, utilizando o método de adaptações heurísticas.

Qualidade	Tempo (s)				
	Adaptação	Colapso	Divisão	*Flipping*	Total
$Q^* = 0.15$	2110 ± 50	920 ± 2	940 ± 50	82.3 ± 0.6	8720 ± 80
$Q^* = 0.20$	2398 ± 9	1055 ± 4	1054 ± 5	95.0 ± 0.8	9010 ± 50
$Q^* = 0.25$	2700 ± 10	1190 ± 6	1187 ± 4	106.4 ± 0.2	9300 ± 30
$Q^* = 0.30$	3230 ± 20	1424 ± 8	1421 ± 5	127.9 ± 0.4	9800 ± 30

Tabela 6.2: Tempo de execução das operações de adaptação em função da qualidade imposta.

6.1.2 Resultados com o algoritmo de adaptação por otimização discreta

Aplicando o método de otimização discreta no algoritmo de evolução temporal, foi possível observar algumas desvantagens de tal método, como mínimos locais. Na figura 6.5 mostramos o resultado da qualidade dos elementos ao longo da simulação, utilizando o critério de qualidade de $Q^* = 0.30$. É importante lembrar que para este tipo de adaptação utilizamos uma medida de qualidade que leva em consideração não só a forma, mas também o campo de tamanhos, como mostrado na equação 5.2.

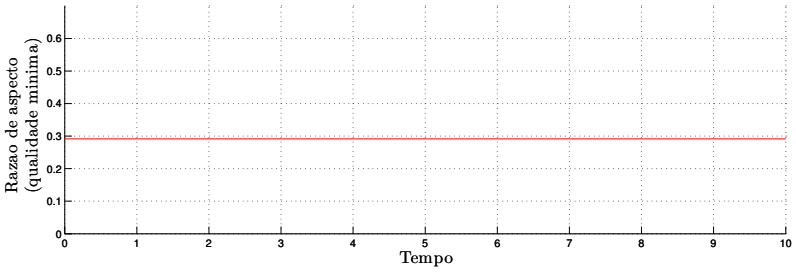


Figura 6.5: Qualidade mínima da malha nos primeiros 10 segundos simulados, utilizando o método de otimização discreta. Note que a qualidade mínima permanece estacionária abaixo de $Q^* = 0.30$, devido a alguns elementos que não podem ser reparados.

Como pode ser visto, o método não foi capaz de satisfazer ao critério de qualidade. Observando as malhas de saída, foi possível perceber que tal efeito é causado por mínimos locais para todos os critérios de qualidade testados entre $Q^* = 0.15$ a $Q^* = 0.30$, como o caso mostrado a seguir.

Em um dado momento a malha possui apenas poucas células de qualidade inferior ao critério estabelecido. Olhando para a figura 6.6, visualmente uma opção para melhorar tal região seria aplicar a operação de *flip* sobre as arestas em destaque. Porém, após a operação, as arestas desta região seriam bem menores do que o determinado pelo campo de tamanhos, o que levaria a medida de qualidade a um valor menor que o valor anterior à operação ser realizada. O método então opta por não realizar o *flip*, permanecendo em um mínimo local.

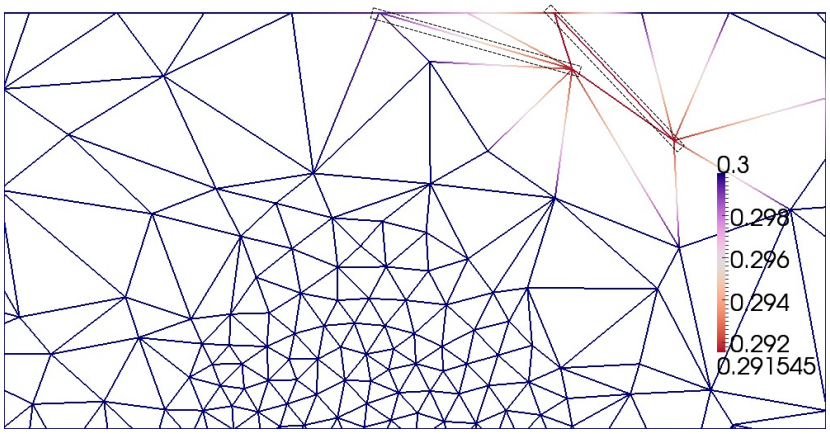


Figura 6.6: Região da malha que apresenta mínimos locais.

Além de não atingir o critério estabelecido, tais mínimos locais causam um custo computacional maior ao método, pois a cada passo de tempo o método de otimização discreta executa o processamento dos elementos da malha e não obtém melhoria.

6.2 Simulação de ascensão de bolhas

Nesta seção apresentamos uma simulação do modelo de ascensão de uma bolha de fluido imerso em um fluido mais denso. Partimos de uma domínio retangular e uma bolha de raio r dados a seguir, conforme mostrado na figura 6.7, e iteramos no tempo de acordo com o modelo descrito na seção 4.2, utilizando os seguintes parâmetros de acordo com o modelo eE:

$$
\begin{aligned}
\delta t &= 1 \times 10^{-4}\,\mathrm{s} \\
t_{final} &= 0.4\,\mathrm{s} \\
N_{mesh} &= 10 \\
g &= 9.806\,\mathrm{m/s^2} \\
\rho_L &= 1.0\,\mathrm{kg/m^3}; \\
\rho_F &= 1 \times 10^3\,\mathrm{kg/m^3}; \\
V_T &= 2.0\,\mathrm{m/s}; \\
\tau &= 0.2\,\mathrm{s}; \\
\mu_L &= 1 \times 10^{-6}\,\mathrm{Pa\,s}; \\
\sigma &= 10\,\mathrm{N/m};
\end{aligned}
$$

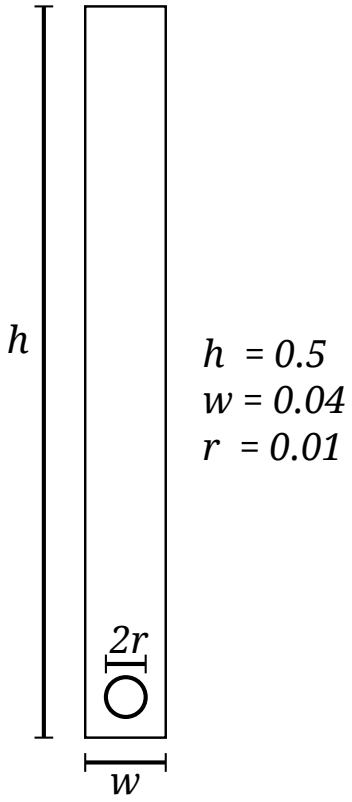
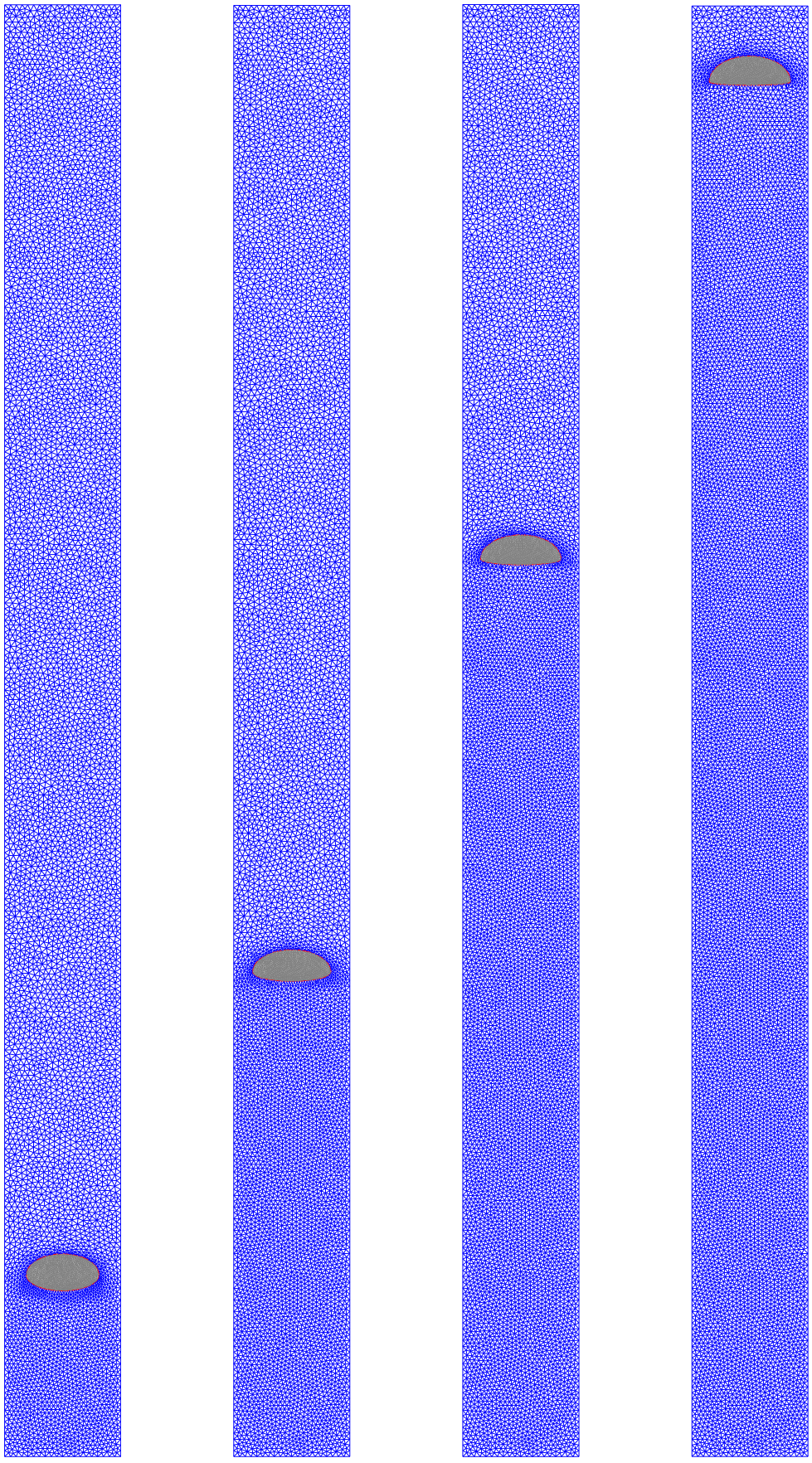


Figura 6.7: Domínio da simulação de ascensão de bolhas.

Para esta simulação utilizamos apenas o método de adaptação heurística, com um campo de tamanhos com $L_{min} = 0.66\,\mathrm{mm}$, $L_{max} = 2.64\,\mathrm{mm}$ e $L_{range} = 6.6\,\mathrm{mm}$.

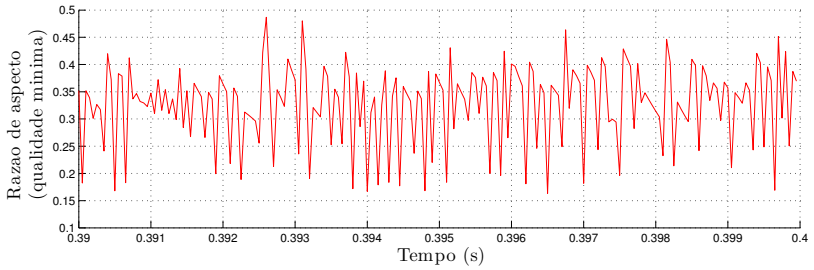
Na figura 6.8 mostramos o resultado da simulação de ascensão de uma bolha de fluido. Como podemos ver nesta simulação a superfície da bolha se deforma, e adaptações sobre a interface da bolha são necessárias, o que como descrito anteriormente pode levar a mudanças de massa das regiões. Para evitar tal efeito aplicamos o método de conservação de massa descrito na seção 3.4 e medimos a área ocupada por cada fluido, onde foi possível constatar que não houve alterações nas áreas após as adaptações, a menos de erros de arredondamento.

Assim como na primeira simulação o método foi capaz de satisfazer aos critérios estabelecidos, como podemos ver pela figura 6.9 e mostra a qualidade média dos elementos da malha bastante elevada, sempre em torno de 0.97.

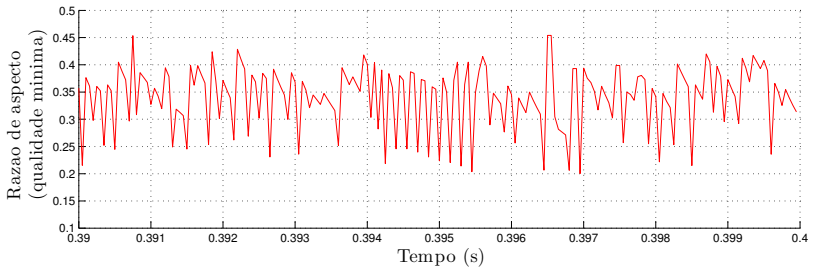


(a) $t = 0.1\,\mathrm{s}$ (b) $t = 0.2\,\mathrm{s}$ (c) $t = 0.3\,\mathrm{s}$ (d) $t = 0.4\,\mathrm{s}$

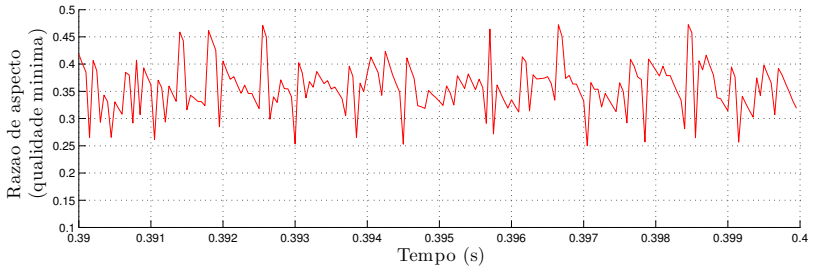
Figura 6.8: Simulação de ascensão de bolhas.



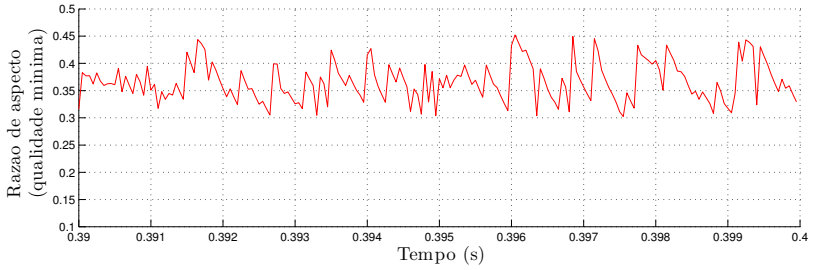
(a) $Q^* = 0.15$



(b) $Q^* = 0.20$



(c) $Q^* = 0.25$



(d) $Q^* = 0.30$

Figura 6.9: Qualidade mínima da malha no último centésimo de segundo simulado, utilizando o método de adaptações heurísticas.

CAPÍTULO

7

Considerações Finais

Ao longo deste trabalho apresentamos e discutimos diversos problemas que são enfrentados ao se trabalhar com métodos de simulação numérica baseados em discretizações por malhas, os quais muitas vezes limitam o potencial de diversas aplicações. Fornecemos um arcabouço metodológico para auxiliar em tais desafios.

Desenvolvemos um pacote de algoritmos e rotinas para lidar com alguns problemas, implementando técnicas de controle e otimização da qualidade dos elementos da malha, técnicas de adaptação da interface entre fluidos com esquemas de conservação de massa, técnicas de mudanças topológicas e preservação de propriedades materiais, além de uma comunicação facilitada destas rotinas com códigos de simulação numérica de escoamentos multifásicos existentes.

Os resultados obtidos demonstram a capacidade das implementações desenvolvidas sobre a estrutura de dados de malhas adaptativas, mantendo a malha com boa qualidade ao tratar grandes deformações com um método de simples implementação.

Foram estudados dois métodos de adaptação de malhas dinâmicas baseados em operações locais: o método de adaptação heurístico e o método de otimização discreta, com objetivo de analisar e comparar o comportamento destes em simulações com superfícies sujeitas a grandes deformações. Com base nos resultados obtidos por simulações de fenômenos de pseudo-física, observamos que o método de adaptação heurístico se mostrou melhor que o método de otimização discreta no controle da qualidade da malha, mesmo não possuindo garantias teóricas de melhorias de qualidade na sua aplicação. A metodologia desenvolvida

neste trabalho foi capaz de manter a razão de aspecto das células da malha acima dos critérios estabelecidos, melhorando a qualidade após deformações consideráveis e mantendo uma representação fiel das interfaces.

Podemos citar como as principais contribuições deste trabalho: a extensão da estrutura de dados de malhas da biblioteca FEPiC++, desenvolvida no ICMC-USP, possibilitando a esta trabalhar com malhas dinâmicas em $2D$; o desenvolvimento de métodos de simples implementação para solucionar problemas presentes na área de tratamento de malhas dinâmicas, como tratamento de superfícies e comunicação de módulos computacionais; e uma análise de desempenho computacional das técnicas de adaptação estudadas, o que se estabelece como uma referência útil para trabalhos futuros em CFD.

Propomos como trabalhos futuros as seguintes atividades: extensão da estrutura de dados de malhas da biblioteca FEPiC++ para tratamento de malhas dinâmicas em $3D$; estudo de maneiras eficientes para definição de campos de tamanhos adequados para simulações multifásicas; e pesquisa de maneiras para tratamento de mínimos locais no método de otimização discreta.

Referências

[1] ALMEIDA, R. C.; FEIJÓO, R. A.; GALEÃO, A. C.; PADRA, C.; SILVA, R. S. Adaptive finite element computational fluid dynamics using an anisotropic error estimator. *Computer Methods in Applied Mechanics and Engineering*, v. 182, n. 3, p. 379–400, 2000.

[2] ALUMBAUGH, T. J.; JIAO, X. Compact Array-Based Mesh Data Structures. *14th International Meshing Roundtable*, v. Proceedings, 2005.

[3] BABUŠKA, I.; AZIZ, A. K. On the angle condition in the finite element method. *SIAM Journal on Numerical Analysis*, v. 13, n. 2, p. 214–226, 1976.

[4] BABUŠKA, I.; MILLER, A. A feedback finite element method with a posteriori error estimation: Part i. the finite element method and some basic properties of the a posteriori error estimator. *Computer Methods in Applied Mechanics and Engineering*, v. 61, n. 1, p. 1–40, 1987.

[5] BABUVŠKA, I.; RHEINBOLDT, W. C. Error estimates for adaptive finite element computations. *SIAM Journal on Numerical Analysis*, v. 15, n. 4, p. 736–754, 1978.

[6] BONITO, A.; NOCHETTO, R. H.; PAULETTI, M. S. Geometrically consistent mesh modification. *SIAM Journal on Numerical Analysis*, v. 48, n. 5, p. 1877–1899, 2010.

[7] BROCHU, T.; BRIDSON, R. Robust topological operations for dynamic explicit surfaces. *SIAM Journal on Scientific Computing*, v. 31, n. 4, p. 2472–2493, 2009.

[8] BUSCAGLIA, G. C.; DARI, E. A. Anisotropic mesh optimization and its application in adaptivity. *International Journal for Numerical Methods in Engineering*, v. 40, n. 22, p. 4119–4136, 1997.

[9] CELATA, G. P.; CUMO, M.; D'ANNIBALE, F.; TOMIYAMA, A. The wake effect on bubble rising velocity in one-component systems. *International journal of multiphase flow*, v. 30, n. 7, p. 939–961, 2004.

[10] CIRILO, E. R. *Modelagem matemática e simulação numérica do transporte de metano em reservatórios de hidrelétricas*. Tese de Doutoramento, 2012.

[11] COMPERE, G.; REMACLE, J.-F.; JANSSON, J.; HOFFMAN, J. A mesh adaptation framework for dealing with large deforming meshes. *International journal for numerical methods in engineering*, v. 82, n. 7, p. 843–867, 2010.

[12] COOK, B. K.; NOBLE, D. R.; WILLIAMS, J. R. A direct simulation method for particle-fluid systems. *Engineering Computations*, v. 21, n. 2/3/4, p. 151–168, 2004.

[13] DOCUMENTATION, C. The cfd general notation system standard interface data structures. *URL: http://www. grc. nasa. gov/WWW/cgns/beta/sids/sids. pdf*, 2007.

[14] DONEA, J.; HUERTA, A.; PONTHOT, J.-P.; RODIGUEZ-FERRAN, A. Arbitrary Lagrangian-Eulerian Methods. *Encyclopedia of Computational Mechanics, E. Stein, páginas 413–438*, v. Fundamentals, 2004.

[15] EDELSBRUNNER, H. *Geometry and topology for mesh generation*. Cambridge University Press, 2001.

[16] FREY, P.; GEORGE, P.-L. *Mesh generation*, v. 32. John Wiley & Sons, 2010.

[17] GEUZAINE, C.; REMACLE, J.-F. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering*, v. 79, n. 11, p. 1309–1331, 2009.

[18] GUENNEBAUD, G.; JACOB, B.; ET AL. Eigen v3. http://eigen.tuxfamily.org, 2010.

[19] HUA, J.; LOU, J. Numerical simulation of bubble rising in viscous liquid. *Journal of Computational Physics*, v. 222, n. 2, p. 769–795, 2007.

[20] KNEPLEY, M. G.; KARPEEV, D. A. Flexible representation of computational meshes. *Preprint ANL/MCS-P1295-1005, Argonne National Laboratory*, 2005.

[21] LOGG, A. Efficient representation of computational meshes. *International Journal of Computational Science and Engineering*, v. 4, n. 4, p. 283–295, 2009.

[22] MÄNTYLÄ, M. An introduction to solid modeling. 1988.

[23] MESRI, Y.; ZERGUINE, W.; DIGONNET, H.; SILVA, L.; COUPEZ, T. Dynamic parallel adaption for three dimensional unstructured meshes: Application to interface tracking. In: *Proceedings of the 17th International Meshing Roundtable*, Springer, p. 195–212, 2008.

[24] MONTEFUSCOLO, F.; SOUSA, F. S. Métodos numéricos para escoamentos com linhas de contato dinâmicas. 2012.

[25] QUAN, S. Simulations of multiphase flows with multiple length scales using moving mesh interface tracking with adaptive meshing. *Journal of Computational Physics*, v. 230, n. 13, p. 5430–5448, 2011.

[26] QUAN, S.; LOU, J.; SCHMIDT, D. P. Modeling merging and breakup in the moving mesh interface tracking method for multiphase flow simulations. *Journal of Computational Physics*, v. 228, n. 7, p. 2660–2675, 2009.

[27] QUAN, S.; SCHMIDT, D. P. A moving mesh interface tracking method for 3d incompressible two-phase flows. *Journal of Computational Physics*, v. 221, n. 2, p. 761–780, 2007.

[28] SHARDT, O.; DERKSEN, J. Direct simulations of dense suspensions of non-spherical particles. *International Journal of Multiphase Flow*, v. 47, p. 25–36, 2012.

[29] SHEWCHUK, J. What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint). *University of California at Berkeley*, 2002.

[30] SHEWCHUK, J. R. Delaunay refinement mesh generation. 1997.

[31] SOUSA, F. S. *Simulação de escoamentos multifásicos em malhas não estruturadas*. Tese de Doutoramento, 2005.